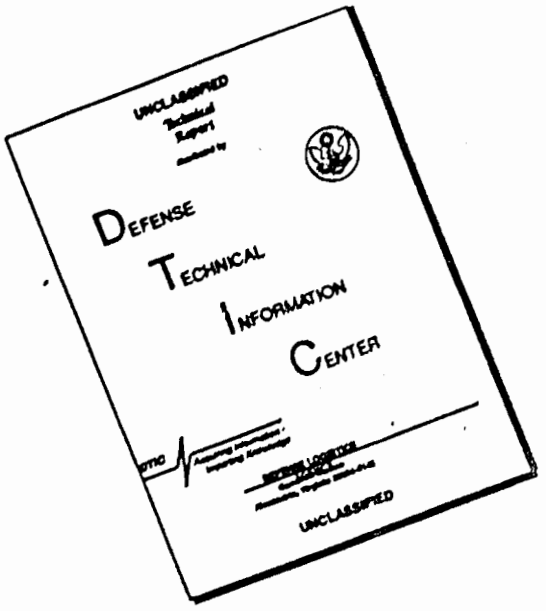
DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

AFOSR-TR- 78-1246

CMU-CS-78-117

LEVEL II

HEARSAY-II:

Tutorial Introduction
&
Retrospective View

Erman & Lesser / Lesser & Erman

May, 1978

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR- 78-1246	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): HEARSAY-II: A TUTORIAL INTRODUCTION and RETROSPECTIVE VIEW.	5. TYPE OF REPORT & PERIOD COVERED: Interim rept.
6. PERFORMING ORG. REPORT NUMBER: CMU-CS-78-117	
7. AUTHOR(s): Erman & Lesser/Lesser & Erman	8. CONTRACT OR GRANT NUMBER(s): F44620-73-C-0074
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University, Department of Computer Science, Pittsburgh, Pennsylvania 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61101E, A02466/7
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, Virginia 22209	12. REPORT DATE: May 1978 13. NUMBER OF PAGES: 46
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

12 49p.

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

10 Lee D. Erman; Victor R. Lesser

18. SUPPLEMENTARY NOTES

16 2466 17 07

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Hearsay-II system, developed at CMU as part of the five-year ARPA speech-understanding project, was successfully demonstrated at the end of that project in September, 1976. This report reprints two Hearsay-II papers which describe and discuss that version of the system:

1. Erman & Lesser. "The Hearsay-II System: A Tutorial", and Chap. 16 in W.A. Lea (ed.) Trends in Speech Recognition, Prentice-Hall, Englewood Cliffs,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data

403 084

NJ, 1978 (in press). (Copy-written by Prentice-Hall -- reprinted here with their kind permission.)

2. Lesser & Erman. "A Retrospective View of the Hearsay-II Architecture." Proc. Inter. Joint Conf. on Artificial Intelligence - 1977, Cambridge, MA, 790-800.

The first paper presents a short introduction to the general Hearsay-II structure and describes the September 1976 configuration of knowledge-sources; it includes a detailed description of an utterance being recognized. The second paper discusses the general Hearsay-II architecture and some of the crucial problems encountered in applying that architecture to the problem of speech understanding.

CMU-CS-78-117

HEARSAY-II:

Tutorial Introduction
&
Retrospective View

Erman & Lesser / Lesser & Erman

May, 1978

The Hearsay-II system, developed at CMU as part of the five-year ARPA speech-understanding project, was successfully demonstrated at the end of that project in September, 1976. This report reprints two Hearsay-II papers which describe and discuss that version of the system:

1. Erman & Lesser. "The Hearsay-II System: A Tutorial", Chap. 16 in W. A. Lea (ed.) *Trends in Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1978 (in press). (Copy-written by Prentice-Hall -- reprinted here with their kind permission.)
2. Lesser & Erman. "A Retrospective View of the Hearsay-II Architecture", *Proc. Inter. Joint Conf. on Artificial Intelligence - 1977*, Cambridge, MA, 790-800.

The first paper presents a short introduction to the general Hearsay-II structure and describes the September 1976 configuration of knowledge-sources; it includes a detailed description of an utterance being recognized. The second paper discusses the general Hearsay-II architecture and some of the crucial problems encountered in applying that architecture to the problem of speech understanding.

For a more general view of Hearsay-II as a structure for problem-solving, see:

3. Erman & Lesser. "A Multi-Level Organization for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge", *Proc. 4th Inter. Joint Conf. on Artificial Intelligence*, Tbilisi, Georgia, USSR, Sept., 1975, 483-490.

For a description of the implementation, see

4. Erman & Lesser. "System Engineering Techniques for Artificial Intelligence Systems", A. Hanson and E. Riseman (eds.) *Computer Vision Systems*, Academic Press, Inc., NY, 1978 (in press). (Also available as CMU Technical Report, Dec., 1977.)

These four papers present a comprehensive overview of the Hearsay-II project and contain pointers to other relevant papers.

Authors' current addresses:

Lee D. Erman: USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90291.

Victor R. Lesser: Computer & Information Sciences Dept., University of Mass., Amherst, MA 01003.

This work was supported by the Defense Advanced Research Projects Agency (F44620-73-C-0074) and is monitored by the Air Force Office of Scientific Research.

THE HEARSAY-II SPEECH UNDERSTANDING SYSTEM: A TUTORIAL

Lee D. Erman and *Victor R. Lesser*

INTRODUCTION

In 1971-72, the Hearsay-I speech understanding system was developed at Carnegie-Mellon University -- the first of a series of such systems. Hearsay-I [Reddy Erman & Neely 73, Reddy Erman Fennell & Neely 73] was a successful attempt to solve the problem of machine understanding of speech in specialized task domains. In this early system, the size of the vocabulary (fewer than 100 words) and complexity of the grammar were very limited. Experiences with Hearsay-I led to the more generalized Hearsay-II architecture [Lesser Fennell Erman & Reddy 75, Erman & Lesser 75, Lesser & Erman 77] in order to handle more difficult problems (e.g., larger vocabularies and less-constrained grammars).

The active development of Hearsay-II extended over three years. During this period, a number of different knowledge-source configurations were constructed within the Hearsay-II framework. The most important of these are called configurations C0 (January, 1975), C1 (January, 1976), and C2 (September, 1976). This last configuration was very successful: it came close to the original ARPA performance goals set out in 1971 to be met by the end of 1976 [Newell et al, 73]. Its performance in September, 1976, was 90% correct semantic interpretation of sentences over a 1011-word vocabulary and constrained syntax [CMU 77].

This presentation is divided into three major sections. First, the Hearsay-II system architecture is presented. The next section discusses in detail the C2 configuration -- the particular types of knowledge that are contained in this configuration, and how this knowledge interacts in order to recognize spoken utterances. The last section contains a detailed example of C2 recognizing an utterance.

THE HEARSAY-II ARCHITECTURE

The Hearsay-II architecture is based on the view that the inherently errorful nature of processing connected speech can be handled only through the effective and efficient cooperation of multiple, diverse sources of knowledge. Additionally, the experimental approach needed for system development requires the ability to add and replace sources of knowledge and to explore different control strategies. Thus, such changes must be relatively easy to accomplish; there must also be ways to evaluate the performance of the system in general and the roles of the various sources of knowledge and control strategies in particular. This ability to experiment conveniently with the system is especially crucial because the amount of knowledge is large and many people are needed to introduce and validate it.

A major focus of the design of the Hearsay-II system was the development of a framework for experimenting with the representation of and cooperation among these diverse sources of knowledge. Based on our experiences with Hearsay-I, we expected to need types

of knowledge and interaction patterns whose details could not be anticipated at the outset of the project. Therefore, instead of designing a specific speech understanding system, we considered Hearsay-II as a model for a class of systems and a framework within which specific configurations of that general model could be constructed and studied. One can think of Hearsay-II as a high-level system for programming speech understanding systems of a certain type -- i.e., those that conform to the Hearsay-II model.

In the Hearsay-II architecture, each of the diverse types of knowledge needed to solve the speech problem is encapsulated in a knowledge source (KS). For speech understanding, typical KSs incorporate information about syntax, semantics, acoustic-phonetics, prosodics, syllabification, coarticulation, etc. The C2 configuration has about ten KS modules. KSs are kept separate, anonymous, and as independent as possible, in order to make the creation, modification, and testing of KS modules as easy as possible.

As one knowledge source makes errors and creates ambiguities, other KSs must be brought to bear to correct and clarify those actions. This KS cooperation should occur as soon as possible after the introduction of an error or ambiguity in order to limit its ramifications. The mechanism used for providing this high degree of cooperation is the hypothesize-and-test paradigm. In this paradigm, solution-finding is viewed as an iterative process. Two kinds of KS actions occur: 1) the creation of an hypothesis, an "educated guess" about some aspect of the problem (e.g., that a particular word was spoken during a specified portion of the utterance), and 2) tests of the plausibility of some hypothesis or sets of hypotheses. For both of these steps, the KS uses a priori knowledge about the problem, as well as the previously generated hypotheses. This "iterative guess-building" terminates when some subset of the hypotheses generated describes the spoken utterance "well-enough" to satisfy some halting criteria.

The Blackboard

The requirement that knowledge sources be independent implies that the functioning (and very existence) of each must not be necessary or crucial to the others. On the other hand, the KSs are required to cooperate in the iterative guess-building, using and correcting one another's guesses; this implies that there must be interaction among the KSs. These two opposing requirements have led to a design in which each KS interfaces to the others externally in a uniform way that is identical across KSs and in which no knowledge source knows which or how many other KSs exist. The interface is implemented as a dynamic global data structure, called the blackboard.

The blackboard is partitioned into distinct information levels (e.g., "phrase", "word", "syllable", and "phone"); each level holds a different representation of the problem space. The current state of problem solution is represented in terms of hypotheses on the blackboard. An hypothesis is an interpretation of a portion of the spoken utterance at a particular level (e.g., an hypothesis might be that the word 'today' occurred from millisecond 100 to millisecond 600 in the utterance). All hypotheses, no matter what their level, have a uniform attribute-value structure. For example, each hypothesis has attributes containing its level, begin- and end-time within the utterance (which can include notions of fuzziness), and plausibility ratings. The level and time attributes place a two-dimensional structure on hypotheses which partitions the blackboard and can be used for addressing hypotheses. Note that two or more hypotheses at the same level with significantly overlapping times are competitors; i.e., they represent competing interpretations of a portion of the utterance.

Hypotheses at different levels are connected through an and/or directed graph structure. Through these connections, hypotheses at each level can be described approximately as abstractions of hypotheses at the next lower level. A partial solution (i.e., a group of hypotheses) at one level can be used to constrain the search at an adjacent level. For example, consider a KS which can predict and rate words based on acoustic information and another KS which knows about the grammar of the language. The first KS can generate a set of candidate word-hypotheses. The second KS can use these hypotheses to generate phrase hypotheses which can be used, in turn, to predict words likely to precede or follow. These predictions can now constrain the search for the first KS.

Knowledge-Source Activation

Each knowledge source is activated in a data-directed manner, based on the occurrence on the blackboard of patterns of hypotheses specific to its interests. For example, a KS which knows how to make hypotheses about words given hypotheses about syllables is activated whenever any KS creates new syllable hypotheses. Once activated, a KS may examine the blackboard, typically in the vicinity of the hypotheses that activated it. Based on its knowledge, the KS may then modify those hypotheses or other hypotheses, or create new hypotheses. Such actions establish new patterns on the blackboard; these patterns may cause other KSs to be activated. This mechanism for KS activation implements a data-directed form of the hypothesize-and-test paradigm.

Each KS has two major components: a precondition and an action. The purpose of the precondition is to find a subset of hypotheses that is appropriate for action by the KS and to invoke the KS on that subset; the subset is called the stimulus frame of the KS instantiation. For example, the precondition of the KS that generates word hypotheses based on syllables looks for new syllable hypotheses. To keep from having to fire continuously to search the blackboard, each precondition declares the primitive kinds of blackboard changes in which it is interested. Each precondition is triggered only when such primitive changes occur (and is then given pointers to all of them). Whenever a precondition is executed, it checks all blackboard events in which it is interested that have occurred since the last time it was executed. For example, a "new hypothesis" to an executing precondition is any hypothesis which was created since the last time the precondition was executed.

The action part of a KS is a program for applying the knowledge to the stimulus frame and making appropriate changes to the blackboard. A stylized description of the likely action that the KS instantiation will perform (if and when it is allowed to execute) is called the response frame. For example, a response frame for the syllable-based word hypothesizer indicates that the action will be to generate hypotheses at the word level and in a time area that includes at least that of the stimulus frame. The stimulus and response frames, which are generated by the precondition component of the KS, provide information for comparing the desirability of execution of a KS instantiation to that of other KS instantiations; this information is used for the scheduling of KS instantiations.

Scheduling of Knowledge-Sources

At any point, there are, in general, a number of pending tasks to execute -- both invoked knowledge sources and triggered preconditions. (In practice, the number of pending tasks often exceeds 200.) If very, very large amounts of processing power (and memory)

were available, one could consider actually activating all KSs in all their possible contexts. This would expand the blackboard with many (competing) hypotheses. Assuming this would eventually terminate (i.e., at some point no new contexts are created), a decision process could then try to pick from all the competing hypotheses that subset which best describes the data -- this would be the system's "solution" to the problem. Because of this combinatoric explosion of possibilities (caused mostly by the problems of variability and incompleteness in the signal and errorfulness of the KSs), this complete expansion is not feasible. Therefore, the control strategy can pick only a small subset of the applicable KS activations; this can be thought of as exploring a limited portion of the (potential) fully-expanded blackboard.

This selection process is implemented by a scheduler which calculates a priority for each waiting task and selects for execution the task with the highest priority. The priority calculation attempts, based on the specific stimulus and response frames of the actions, to estimate the usefulness of the action in fulfilling the overall system goal of recognizing the utterance. A more detailed explanation of the scheduler is contained in the next section and in [Hayes-Roth & Lesser 77].

The Hearsay-II Implementation

Based on the architecture just described, a high-level programming system was constructed to provide an environment for programming knowledge sources, configuring groups of them into systems, and executing them. Because KS interactions occur via the blackboard (triggering on patterns, accessing hypotheses, and making modifications) and the blackboard has a uniform structure, KS interactions are also uniform. Thus, one set of facilities can serve all KSs. Facilities are provided for

- o defining the levels on the blackboard,
- o configuring groups of KSs into runnable systems,
- o accessing and modifying hypotheses on the blackboard,
- o activating and scheduling KSs.

These facilities, along with other utilities for debugging and user (researcher) interaction, are called the Hearsay-II 'kernel'. The kernel is the high-level environment for creating and testing KSs and configurations of them [Erman & Lesser 78].

Hearsay-II is implemented in the SAIL programming system [Reiser 76], an Algol-60 dialect which has a sophisticated compile-time macro facility as well as a large number of data structures (including lists and sets) and control modes which are implemented fairly efficiently. The Hearsay-II kernel provides a high-level environment for KSs at compile-time by extending SAIL's data types and syntax through declarations of procedure calls, global variables, and complex macros. This extended SAIL provides an explicit structure for the specification of a KS and its interaction with other KSs (through the blackboard). The high-level environment also provides mechanisms that enable KSs to specify to the kernel (usually in non-procedural ways) a variety of information which the kernel uses when configuring a system, scheduling KS activity, and controlling user interaction.

The knowledge in a KS is represented using SAIL data structures and code, in whatever form the KS developer finds appropriate. The kernel environment provides the facilities for structuring the interface between this knowledge and other KSs, via the blackboard. For example, the syntax KS contains a grammar for the specialized task language that is to be recognized; this grammar is in a compact, network form. The KS also contains procedures for searching this network, for example, to parse a sequence of words. The kernel provides

facilities (1) for triggering this KS whenever new word hypotheses appear on the blackboard, (2) for the KS to read those word hypotheses (in order to find the sequence of words to be parsed), and (3) for the KS to create new hypotheses on the blackboard, indicating the structure of the parse.

THE KNOWLEDGE-SOURCES OF SEPTEMBER, 1976

In this section, a description of the September, 1976, version of the Hearsay-II system -- configuration C2 -- is given in terms of the functions and interactions of its knowledge sources. Included is an example run of the system.

The task for the system is to answer questions about and retrieve documents from a collection of computer science abstracts (in the area of artificial intelligence). Example sentences are

"Which abstracts refer to theory of computation?"
"List those articles."
"What has McCarthy written since nineteen seventy-four?"

The vocabulary contains 1011 words (in which each extended form of a root, e.g., the plural of a noun, is counted separately, if it appears). The grammar which defines the legal sentences is context free and includes recursion. The style of the grammar is such that there are many more non-terminals than in conventional syntactic grammars; the information contained in the greater number of nodes provides semantic and pragmatic constraint within the grammatical structure. For example, in place of 'Noun' in a conventional grammar, this grammar includes such non-terminals as 'Topic', 'Author', 'Year', 'Publisher', etc.

The grammar allows each word to be followed, on the average, by seventeen other words of the vocabulary.[1] The standard deviation of this measure is very high (about 51), since some words can be followed by many others (up to 300 in several cases). For the sentences used for performance testing, the average length is seven words and the average number of words that can follow any initial portion of the sentence is thirty-four.

Figure 1 gives a schematic of configuration C2 as it was operational in September, 1976. The levels are indicated by solid horizontal lines and are labeled at the left. KSs are indicated by vertical arcs with the circled end indicating the level where its stimulus frame is and the pointed end indicating the level of its response frame. The name of a KS is connected to its arc by a dashed horizontal line.

Signal Acquisition, Parameter Extraction, Segmentation, Labeling (SEG)

An input utterance is spoken into a medium-quality Electro-Voice RE-51 close-speaking headset microphone in a fairly noisy environment (>65 db). The audio signal is low-passed filtered and 9-bit sampled at 10 KHz. All subsequent processing, as well as controlling the A/D converter, is digital and is done on a time-shared PDP-10 computer. Four parameters (called "ZAPDASH") are derived by simple algorithms operating directly on the sampled signal

1 Actually, a family of grammars was generated, varying in the number of words (terminals) and in the number and complexity of sentences allowed. The grammar described here and used in most of the testing is called "X05".

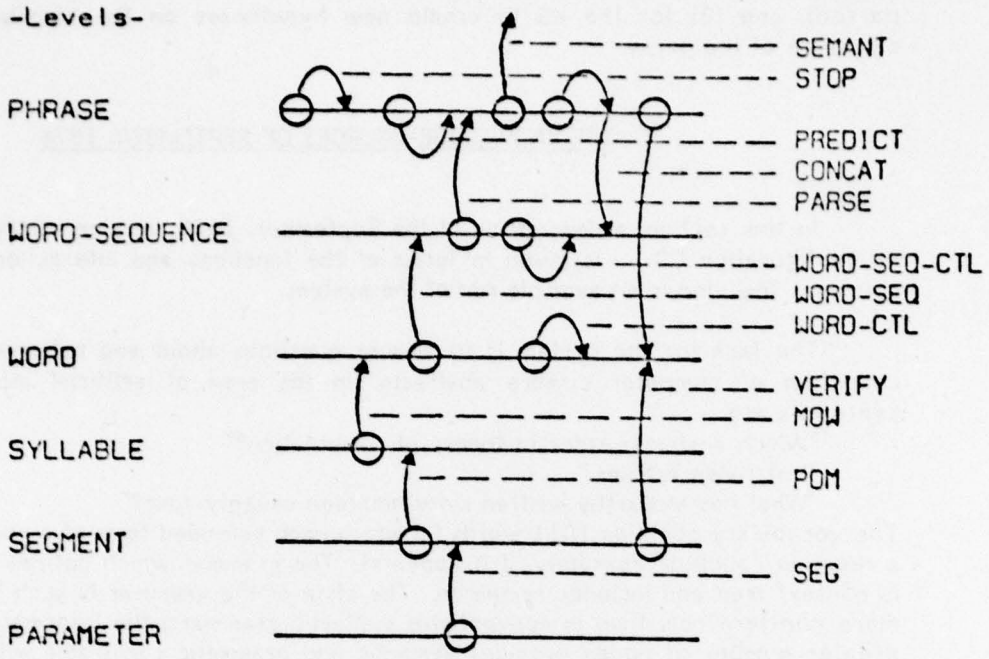


Figure 1. The levels and knowledge sources of configuration C2.

[Goldberg Reddy & Gill 77]. These parameters are extracted in real-time and are used initially to detect the beginning and end of the utterance.

The ZAPDASH parameters are next used by the SEG knowledge-source as the basis for an acoustic segmentation and classification of the utterance. This segmentation is accomplished by an iterative refinement technique: First, silence is separated from non-silence; then, the non-silence is broken down into the sonorant and non-sonorant regions, etc. Eventually, five classes of segments are produced: silence, sonorant peak, sonorant non-peak, fricative, and flap. Associated with each classified segment is its duration, absolute amplitude, and amplitude relative to its neighboring segments (i.e., local peak, local value, or plateau). The segments are contiguous and non-overlapping, with one class designation for each.

Finally, the SEG KS does a finer labeling of each segment. The labels are allophonic-like; there are currently 98 of them. Each of the 98 labels is defined by a vector of auto-correlation coefficients [Itakura 75]. These templates are generated from speaker-dependent training data that have been hand-labeled. The result of the labeling process, which matches the central portion of each segment against each of the templates using the Itakura metric, is a vector of 98 numbers; the i'th number is an estimate of the (negative log) probability that the segment represents an occurrence of the i'th allophone in the label set.

Word Spotting (POM, MOW, WORD-CTL)

The initial generation of words, bottom-up, is accomplished by a three-step process.

First, using the labeled segments as input, the POM knowledge source [Smith 76] generates hypotheses for likely syllable classes. This is done by first identifying syllable nuclei and then "parsing" outward from each nucleus. The syllable-class parsing is driven by a probabilistic "grammar" of "syllable-class -> segment" productions; the rules and their probabilities are learned by an off-line program which is trained on hand-labeled utterances. (The current training, which is speaker-dependent, uses 60 utterances containing about 360 word tokens.) For each nucleus position, several competing syllable-class hypotheses are generated -- typically three to eight.

The syllable classes are used to hypothesize words. Each of the 1011 words in the vocabulary is specified by a pronunciation description. For word hypothesization purposes, an inverted form of the dictionary is kept, in which there is associated with each syllable-class all the words which have some pronunciation containing that syllable-class. The MOW KS [Smith 76] looks up each hypothesized syllable class and generates word candidates from among those words containing that syllable-class. For each word that is multi-syllabic, all of the syllables in one of the pronunciations must match above a threshold. Typically, about 50 words of the 1011-word vocabulary are generated at each syllable nucleus position.[2]

Finally, the generated word candidates are rated and their begin- and end-times adjusted by the WIZARD procedure [McKeown 77]. For each word in the vocabulary, WIZARD has a network which describes the possible pronunciations. This rating is calculated by finding the path through the network which best matches the labeled segments, using the distances associated with each label for each segment; the rating is then based on the difference between this best path and the segment labels.[3]

The result of the processing to this point is a set of words. Each word includes a begin-time, an end-time, and a confidence rating. MOW selects a subset of these words, based on their times and ratings, to be hypothesized; it is these selected word hypotheses that form the base for the "top-end" processing. Typically, these hypotheses include about 75% of the words actually spoken (i.e., "correct" word hypotheses). Each correct hypothesis has a rating which ranks it on the average about three, as compared to the five to twenty-five or so incorrect hypotheses which compete with it (i.e., which significantly overlap it in time). The non-selected words are retained internally by MOW for possible later hypothesization.

The amount of hypothesization that MOW does is controlled by the WORD-CTL ('Word Control') KS. WORD-CTL creates "goal" hypotheses at the word level; these are interpreted by MOW as indicating how many word hypotheses to attempt to create in each time area. One can think of MOW as a generator of word hypotheses (from the candidates it creates internally) and WORD-CTL as embodying the policy of how many to hypothesize. This clear separation of policy from mechanism has facilitated experimentation with various control schemes. For example, a trivial change to WORD-CTL, such that goal hypotheses are generated only at the start of the utterance ("left-hand end"), results in MOW creating word hypotheses only at the start, thus forcing all top-end processing to be left-to-right.

---

2 Since the September, 1976, version, the POM and MOW KSs have been replaced by Noah [Smith 77, Smith & Sambur 78 (section 7-3.2.2.3)]. This KS outperforms POM-MOW on the 1011-word vocabulary (in both speed and accuracy) and is able to handle much larger vocabularies -- it has a performance degradation which is only logarithmic in vocabulary size in the range of 500 to 19,000 words.

3 WIZARD is, in effect, a miniature version of the HARPY speech recognition system [Lowerre 76, Lowerre & Reddy 78], except that it has one network for each word, rather than one network with all words and all sentences.

WORD-CTL fires at the start of processing of an utterance in order to create goal hypotheses. Subsequently, it may re-trigger if the over-all search process stagnates; this condition is recognized as there being no waiting KS instantiations above a certain priority (as described in the section below on "Attention Focussing") or as the global measures of current state of the problem solution not having increased in the last several KS executions.

Top-End Processing

Word-Island Generation (WORD-SEQ, WORD-SEQ-CTL) - The WORD-SEQ knowledge source [Lesser Hayes-Roth Birnbaum & Cronk 77] has the job of generating, from the word hypotheses generated bottom-up, a small set (about three to ten) of word sequence hypotheses. Each of these sequences, or islands, can be used as the basis for expansion into larger islands, hopefully culminating in an hypothesis that spans the entire utterance. Multi-word islands are used rather than single-word islands because of the relatively poor reliability of ratings of single words as well as the limited syntactic constraint supplied by single words.

WORD-SEQ uses two kinds of knowledge to generate multi-word islands:

- A table derived from the grammar indicates for every ordered pair of words in the vocabulary (1011 x 1011) whether that pair can occur in that order in some sentence of the defined language. This binary table (which contains about 1.7% "1"'s) thus defines "language-adjacency".

- Acoustic-phonetic knowledge, embodied in the JUNCT ('juncture') procedure, is applied to pairs of word hypotheses and is used to decide if that pair might be considered to be time-adjacent in the utterance. JUNCT uses the dictionary pronunciations and examines the segments at their juncture (gap or overlap) in making its decision.

WORD-SEQ takes the highest-rated single words and generates multi-word sequences by expanding them with other hypothesized words that are both time- and language-adjacent. This expansion is controlled by heuristics based on the number and ratings of competing word hypotheses. The best of these words sequences (which occasionally includes single words) are hypothesized.

The WORD-SEQ-CTL ('Word-Sequence-Control') KS controls the amount of hypothesization that WORD-SEQ does by creating "goal" hypotheses which are interpreted by WORD-SEQ as indicating how many hypotheses to create. This provides the same kind of separation of policy and mechanism achieved in the MOW/WORD-CTL pair of KSs. WORD-SEQ-CTL fires at the start of processing of an utterance in order to create the goal hypotheses. Subsequently, WORD-SEQ-CTL triggers if stagnation is recognized; it then modifies the word-sequence goal hypotheses, thus stimulating WORD-SEQ to generate new word-sequence islands from which the search may be be more fruitful. WORD-SEQ will generate the additional hypotheses by decomposing word-sequence islands already on the blackboard or by re-generating islands which were initially discarded because their ratings were too low.

Word-Sequence Parsing (PARSE) - Because the syntactic constraint used in the generation of the word sequences is only pair-wise, a sequence longer than two words might not be syntactically acceptable. The PARSE knowledge source of the SASS module [Hayes-Roth

Erman Fox & Mostow 77, Hayes-Roth Mostow & Fox 78] can parse a word sequence of arbitrary length, using the full constraints given by the language. This parsing does not require that the word sequence form a complete non-terminal in the grammar nor that the sequence be sentence-initial or sentence-final -- the words need only occur contiguously somewhere in some sentence of the language. If a sequence hypothesis does not parse, the hypothesis is marked as "rejected". Otherwise, a phrase hypothesis is created. Associated with the phrase hypothesis is the word sequence of which it is composed, as well as information about the way (or ways) the words parsed.

Word Predictions from Phrases (PREDICT) - The PREDICT knowledge source of the SASS module can, for any phrase hypothesis, generate predictions of all words which can immediately precede and all which can immediately follow that phrase in the language. In doing the computation to generate these predictions, this KS uses the parsing information attached to the phrase hypothesis by the parsing component.

Word Verification (VERIFY) - An attempt is made to verify the existence of or reject each such predicted word, in the context of its predicting phrase. This verification is handled by the VERIFY knowledge source. If verified, a confidence rating for the word must also be generated. First, if the word has been hypothesized previously and passes the test for time-adjacency (by the JUNCT procedure), it is marked as verified and the word hypothesis is associated with the prediction. (Note that a single word hypothesis may thus become associated with several different phrases.) Second, a search is made of the internal store created by MOW to see if the candidate can be matched by a previously-generated candidate which had not been hypothesized. Again, JUNCT makes a judgment about time-adjacency. Finally, WIZARD compares its word-pronunciation network to the segments in an attempt to verify the prediction.

For each of these different kinds of verification, the approximate begin-time (end-time) of the word being predicted to the right (left) of the phrase is taken to be the end-time (begin-time) of the phrase. The end-time (begin-time) of the predicted word is not known and, in fact, one requirement of the verification step is to generate an approximate end-time (begin-time) for the verified word. In general, several different "versions" of the word may be generated which differ primarily in their end-times; since no context to the right (left) of the predicted word is given, several different estimates of the end (beginning) of the word may be plausible based solely on the segmental information.

Word-Phrase Concatenation (CONCAT) - For each verified word and its predicting phrase, a new and longer phrase may be generated. This process, accomplished by the CONCAT knowledge source of SASS, which is similar to the PARSE knowledge source, involves parsing the words of the original phrase augmented by the newly verified word. The extended phrase is then hypothesized and includes a rating based on the ratings of the words that compose it.

If a verified word is already associated with some other phrase hypothesis, CONCAT tries to parse that phrase with the predicting phrase. If successful, a new, larger phrase hypothesis is created which represents the merging of the two phrases.

Complete Sentences and Halting Criteria (STOP) - Two unique "word" hypotheses are

generated before the first and after the last segment of the utterance to denote begin and end of utterance, respectively. These same "words" are included in the syntactic specification of the language and appear as the first and last terminals of every complete sentence. Thus, any verified phrase that includes these as its extreme constituents is a complete sentence and spans the entire utterance. Such a sentence becomes a candidate for selection as the system's recognition result.

In general, the control and rating strategies do not guarantee that the first such complete spanning hypothesis found will have the highest rating of all possible spanning sentence hypotheses that might be found if the search were allowed to continue, so the system does not just stop with the first one generated. However, the characteristics of such an hypothesis are used by the STOP knowledge source to prune from further consideration other partial hypotheses which, because of their low ratings, are unlikely to be extendible into spanning hypotheses with ratings higher than the best already-discovered spanning sentence. This heuristic pruning procedure is based on the form of the ratings function (i.e., how the rating of the phrase is derived from its constituent words). The pruning procedure considers each partial phrase and uses the ratings of other word hypotheses in the time areas not covered by the phrase to determine if the phrase might be extendible to a phrase rated higher than the spanning hypothesis; if not, the partial phrase is pruned. This pruning process and the rating and halting policies are discussed in [Mostow 77].

The recognition processing finally halts in one of two ways: First, there may be no more partial hypotheses left to consider for predicting and extending. Because of the combinatorics of the grammar and the likelihood of finding some prediction that is rated at least above *the absolute rejection threshold, this form of* termination happens when the pruning procedure has been effective and has eliminated all competitors. Second, the expenditure of a predefined amount of computing resources (time or space) also halts the recognition process; the actual thresholds used are set according to the past performance of the system on similar sentences (i.e., of the given length and over the same vocabulary and grammar).

Once the recognition process is halted, a selection of one or more phrase hypotheses is made to represent the result. If at least one spanning sentence hypothesis was found, the highest-rated such hypothesis is chosen; otherwise, a selection of several of the highest-rated of the partial phrase hypotheses is made, biasing the selection to the longest ones which tend to overlap (in time) the least.

Hypothesis Ratings (RPOL) - The RPOL KS runs in high priority immediately after any KS action that creates a new hypothesis or that modifies an existing hypothesis. RPOL uses rating information on the hypothesis, as well as rating information on hypotheses to which the stimulus hypothesis is connected, to calculate the over-all rating of the stimulus hypothesis.

Attention Focussing - The top-end processing operations include (a) word-island generation, (b) word sequence parsing, (c) word prediction from phrases, (d) word verification, and (e) word-phrase concatenation. Of these, (c), (d), and (e) are the most frequently performed. Typically, there are a number of these actions waiting to be performed at various places in the utterance. The selection at each point in the processing of which of these actions to perform is a problem of combinatoric control, since the execution of each action usually generates other actions to be done.

To handle this problem, the Hearsay-II system has a statistically-based scheduler [Hayes-Roth & Lesser 77] which calculates a priority for each action and selects, at each time, the waiting action with the highest priority. The priority calculation attempts to estimate the usefulness of the action in fulfilling the over-all system goal of recognizing the utterance. The calculation is based on the stimulus and response frames specified when the action is triggered. For example, the word verifier is triggered whenever words are predicted from a phrase hypothesis; the information passed to the scheduler in order to help calculate the priority of this instantiation of the verifier includes such things as the time and rating of the predicting phrase (in the stimulus frame) and the number of words predicted (as given in the response frame). In addition to the action-specific information, the scheduler keeps track of the overall state of the system in terms of the kinds and quality of hypotheses in each time area.

Interpretation and Response (SEMANT, DISCO)

The SEMANT knowledge-source [Fox & Mostow 77] accepts the word sequence(s) result of the recognition process and generates an interpretation in an unambiguous format for interaction with the data base that the speaker is querying. The interpretation is constructed by actions associated with "semantically interesting" non-terminals (which have been pre-specified for the grammar) in the parse tree(s) of the recognized sequence(s). If recognition results in two or more partial sequences, SEMANT constructs a consistent interpretation based on all of the partial sentences, taking into account for each partial sentence its rating, temporal position, and semantic consistency, as compared to the other partial sentences.

The DISCO ('discourse') knowledge-source [Hayes-Roth Gill & Mostow 77] accepts the formatted interpretation of SEMANT and produces a response to the speaker. This response is often the display of a selected portion of the queried data base. In order to retain a coherent interpretation across sentences, DISCO has a finite-state model of the discourse which is updated with each interaction.

AN EXAMPLE OF RECOGNITION

Following is a description of the recognition of the utterance "ARE ANY BY FEIGENBAUM AND FELDMAN?" by configuration C2 of Hearsay-II.[4] Each major step of the processing is shown; a step usually corresponds to the action of a knowledge source. Execution of the preconditions is not shown explicitly, nor is indication given of knowledge-source instantiations which are never scheduled. Also, executions of RPOL are not shown.

The name of the KS activated at each step follows the step number. If the KS name is followed by an asterisk, this indicates that the hypotheses in the stimulus frame of this KS instantiation are all correct. Single numbers in parentheses after hypotheses are their ratings (on a scale of 0-100). All times given are in centi-second units; thus the duration of the whole utterance, which was 2.25 seconds, is 225. When begin- and end-times of hypotheses are given, they appear as two numbers separated by a colon (e.g., 52:82). Hypotheses which are correct are marked with an asterisk.

4 For reasons of clarity, the description differs from the actual run in a few details.

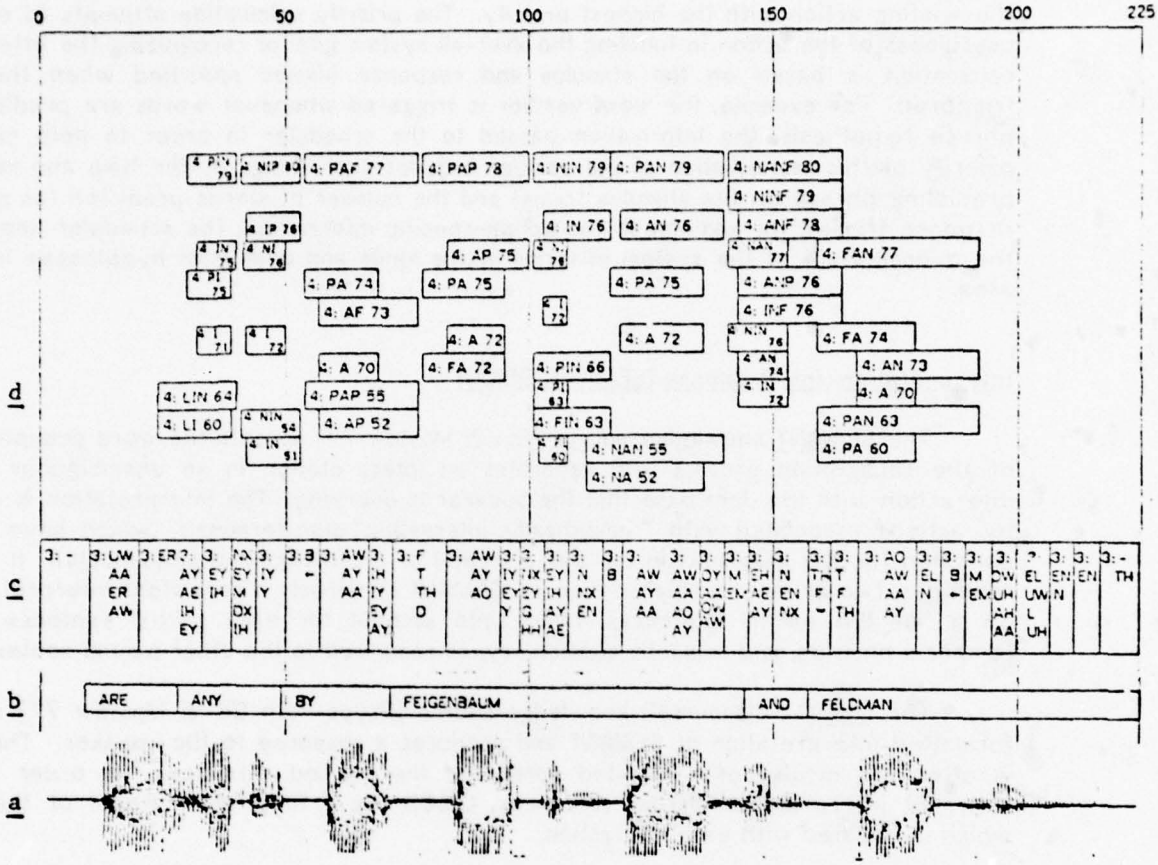


Figure 2.d. Syllable-Classes.
Figure 2.c. Segments.
Figure 2.b. The correct words (for reference).
Figure 2.a. The waveform of "Are any by Feigenbaum and Feldman?".

Fig. 2. The example utterance.

The waveform of the spoken utterance is shown in Fig. 2.a. The "correct" word boundaries (determined by human inspection) is shown in Fig. 2.b for reference. The remaining sections of Fig. 2 contain all the hypotheses created by the KSs on the blackboard (except that the goal hypotheses created by WORD-CTL and WORD-SEQ-CTL are not shown). Each hypothesis is indicated by a box; the hypotheses are grouped by level -- segment, syllable, word, word-sequence, and phrase. Within each hypothesis box, the number preceding the colon indicates the step number in which the hypothesis was created. The symbol following the colon names the hypothesis. At the word level and above, a "*" following the symbol indicates that the hypothesis is correct. The trailing number within the hypothesis box is the rating, on a scale of 0 (lowest) to 100.

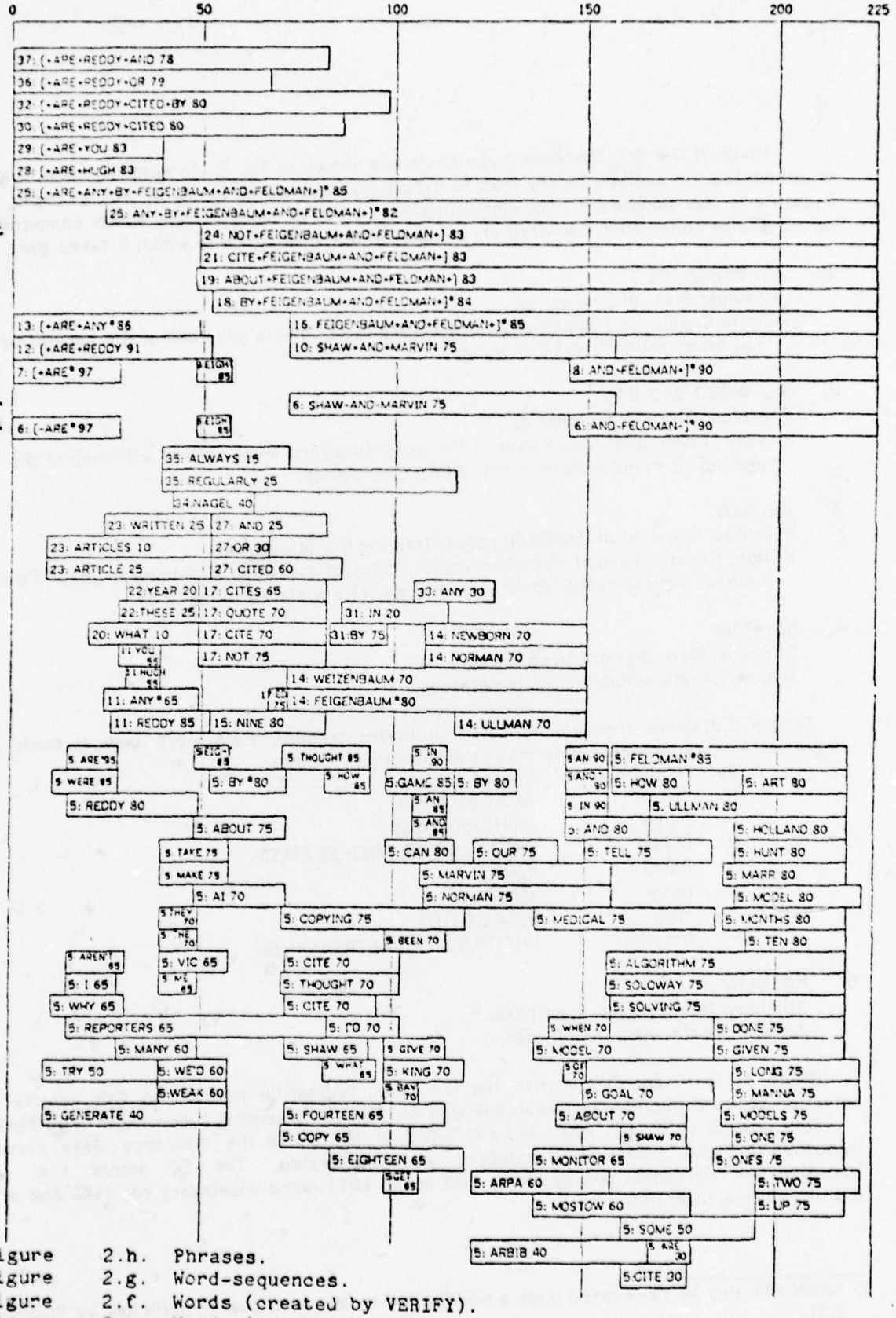


Figure 2.h. Phrases.
Figure 2.g. Word-sequences.
Figure 2.f. Words (created by VERIFY).
Figure 2.e. Words (created by MOW).

Fig. 2 (continued)

None of the links between hypotheses are shown in Fig. 2. In general, each hypothesis is connected via multiple binary links to hypotheses above and below it. For example, a word hypothesis has downward links connecting it to each of the syllables which compose it and upward links connecting it to each phrase and/or word-sequence in which it takes part.

1. KS: WORD-CTL
   Stimulus: Start of processing.
   Action: Create goal hypotheses at the word level. These will control the amount of hypothesization that MOW will do.

2. KS: WORD-SEQ-CTL
   Stimulus: Start of processing.
   Action: Create goal hypotheses at the word-sequence level. These will control the amount of hypothesization that WORD-SEQ will do.

3. KS: SEG
   Stimulus: Creation of ZAPDASH parameters for the utterance.
   Action: Create segment hypotheses with vector of estimated allophone probabilities. (The several highest-rated labels of each segment are shown in Fig. 2.c.)

4. KS: POM
   Stimulus: New segment hypotheses.
   Action: Create syllable-class hypotheses.

Figure 2.d shows the syllable-class hypotheses created. Each class name is made up of single-letter codes representing classes of phones, as follows:

Code	Phone-class	Phones in class
A	A-like	AE,AA,AH,AO,AX
I	I-like	IY,IH,EY,EH,IX,AY
U	U-like	OW,UH,U,UW,ER,AW,OY,EL,EM,EN
L	liquid	Y,W,R,L
N	nasal	M,N,NX
P	stop	P,T,K,B,D,G,DX
F	fricative	HH,F,TH,S,SH,V,DH,Z,ZH,CH,JH,WH

5. KS: MOW
   Stimulus: New syllable hypotheses.[5]
   Action: Create word hypotheses.

Steps 1, 3, 4, and 5 comprise the low level, bottom-up processing; this results in a selection of word hypotheses (created in step 5). Figure 2.e depicts these word hypotheses.

Four words (ARE, BY, AND, and FELDMAN) of the six in the utterance were correctly hypothesized; 86 incorrect hypotheses were generated. The 90 words that were hypothesized represents approximately 1.5% of the 1011-word vocabulary for each one of the 6 words in the utterance.

---

5 MOW will also be re-invoked upon a modification to the word goal hypotheses by WORD-CTL.

6. KS: WORD-SEQ
   Stimulus: New words created bottom-up.
   Action: Create 4 word-sequence hypotheses: AND-FELDMAN-]*(90,145:225), [-ARE*(97,0:28), SHAW-AND-MARVIN(75,72:157), EIGHT(85,48:57).

Step 6 results in the generation of 4 multi-word sequences. (See Fig. 2.g.) These will be used as initial, alternative anchor points for additional searching. Note that two of these islands are correct, each representing an alternative search path that potentially can lead to a correct interpretation of the utterance.

In earlier versions of KS configuration of the system (e.g., C1), low-level processing was not done in the serial, lock-step manner as in steps 3, 4, and 5 (i.e., level-to-level, where each level is completely processed before processing on the next higher level is begun). Rather, processing was done in an asynchronous, data-directed manner (i.e., as interesting hypotheses were generated at one level, they were immediately propagated to and processed by KSs operating at higher levels). It was found that the asynchronous processing at these lower levels (e.g., segment, syllable, and word) was inappropriate because there was not enough accuracy in credibility ratings of hypotheses to form hypothesis islands that could direct the search reliably. It is only with the word-sequence hypotheses produced in step 6 that the reliability of the hypothesis ratings is high enough that selective search can be employed. This conclusion is substantiated by experiments with several island-driving strategies [Lesser Hayes-Roth Birnbaum & Cronk 77].

High level processing on the multi-word sequences is accomplished by the following KSs: PARSE, PREDICT, VERIFY, CONCAT, STOP, and WORD-SEQ-CTL. Since an execution of the VERIFY KS will often immediately follow the execution of the PREDICT KS (each on the same hypothesis), we have combined the descriptions of these two KS executions into one step for ease of understanding.

Steps 7 through 10 involve the PARSE KS. The PARSE KS verifies whether a multi-word sequence (created in step 6) is a legal language fragment of the grammar. If the sequence is grammatical, a phrase hypothesis is constructed from it; otherwise, the sequence is marked rejected. In this example, all four multi-word sequences were verified to be valid language fragments. However, if a multi-word sequence had been rejected, the WORD-SEQ KS might be reinvoked to generate additional multi-word sequences in the time area of the rejected sequence. WORD-SEQ would generate the additional hypotheses by decomposing word-sequence islands already on the blackboard or by re-generating islands which were initially discarded because their ratings were too low. Additional word-sequence hypotheses might also be generated in response to the modification of "goal" hypotheses at the word-sequence level by the WORD-SEQ-CTL.

The scheduling strategy is so parameterized that processing at the phrase level is delayed until an adequate number of highly-rated phrase hypothesis islands are generated. This strategy is not built directly into the scheduler, but rather is accomplished (1) by appropriately setting external scheduling parameters (i.e., the high setting of the priorities of WORD-SEQ and PARSE KS actions in contrast to those of PREDICT, VERIFY, and CONCAT),[6] and (2) by taking into account the current state of hypotheses on the phrase level of the blackboard in evaluating the usefulness of potential KS actions as described by their response frames.

7. KS: PARSE*
   Stimulus: [-ARE* (word sequence)
   Action: Create phrase: [+ARE* (97,0:28)

6 These settings are determined empirically by observing a number of training runs. They are not adjusted during test runs of the system.

8. KS: PARSE*
   Stimulus: AND-FELDMAN-]* (word sequence)
   Action: Create phrase: AND+FELDMAN+]* (90,145:225)

9. KS: PARSE
   Stimulus: EIGHT (word sequence)
   Action: Create phrase: EIGHT (85,48:57)

10. KS: PARSE
    Stimulus: SHAW-AND-MARVIN (word sequence)
    Action: Create phrase: SHAW+AND+MARVIN (75,72:157)

Each of the four executions of the PARSE KS (steps 7-10) results in the creation of a phrase hypothesis; each phrase is shown in Fig. 2.h. Each of these hypotheses causes an invocation of the PREDICT KS. The PREDICT KS attempts to extend a phrase hypothesis through the predictions of words that can, according to the grammar, follow or precede the hypothesis. Its action is to attach a "word-predictor" attribute to the hypothesis which specifies the predicted words. Not all of these PREDICT KS instantiations are necessarily executed (and thus indicated as a step in the execution history). For instance, further processing on the phrases [+ARE and AND+FELDMAN+] is sufficiently positive that the scheduler never executes the instantiation of PREDICT for the phrase SHAW+AND+MARVIN (created in step 10). In turn, VERIFY is invoked by the placing of a word-predictor attribute on a phrase hypothesis. For each word on the attribute list that VERIFY verifies (against the segmental data), it creates a word hypothesis (if one does not already exist) and the word is placed on a "word-verification" attribute of the phrase hypothesis. (Such newly-created word hypotheses are shown in Fig. 2.f.) CONCAT is then invoked on phrase hypotheses which have word-verification attributes attached. For each verified word, the phrase and new word are parsed together and a new, extended phrase hypothesis is created (and shown in Fig. 2.h). If all word predictions to the right or left of the phrase had been rejected, the phrase hypothesis is marked as "rejected", as is the underlying word-sequence hypothesis if all the phrase hypotheses it supports are rejected. (Note that this last action will re-trigger WORD-SEQ to generate more word sequences.)

11. KS: PREDICT & VERIFY*
    Stimulus: [+ARE* (phrase)
    Action: Predict (from the grammar) 292 words to right. Reject (using the acoustic information) 277 of them. The four highest-rated of the fifteen verified words are REDDY(85,26:52), ANY*(65,24:49), HUGH(55,30:39), and YOU(55,28:39).

12. KS: CONCAT
    Stimulus: [+ARE* (phrase), REDDY (word)
    Action: Create phrase: [+ARE+REDDY (91,0:52)

13. KS: CONCAT*
    Stimulus: [+ARE* (phrase), ANY* (word)
    Action: Create phrase: [+ARE+ANY* (86,0:49)

In steps 11 through 13, the highly-rated phrase [+ARE is extended and results in the generation of the additional phrases [+ARE+REDDY and [+ARE+ANY. These phrases, however, are not immediately extended because the predicted words REDDY and ANY are not rated sufficiently high. Instead, the scheduler, pursuing a strategy more conservative than strict

best-first, investigates phrases that look almost as good as the best one. This scheduling strategy results in the execution of the PREDICT and VERIFY KSs on two of the other initial phrase islands: AND+FELDMAN+] and EIGHT.

14. KS: PREDICT & VERIFY*
    Stimulus: AND+FELDMAN+]* (phrase)
    Action: Predict 100 words to left. Reject 76 of them. The best of the verified 24 (in descending rating order) are FEIGENBAUM*(80,72:150), WEIZENBAUM(70,72:150), ULLMAN(70,116:150), NORMAN(70,108:150), and NEWBORN(70,108:150)

15. KS: PREDICT & VERIFY
    Stimulus: EIGHT (phrase)
    Action: Predict the word NINE to right and verify it (80,52:82). Predict SEVEN to left, reject prediction.

The attempted extension of the phrase EIGHT at step 15 is not successful -- none of the grammatically predicted words is acoustically verified, even using a lenient threshold. Thus, this phrase is marked rejected and is dropped from further consideration.

16. KS: CONCAT*
    Stimulus: FEIGENBAUM* (word), AND+FELDMAN+]* (phrase)
    Action: Create phrase: FEIGENBAUM+AND+FELDMAN+]* (85,72:225)

Beginning with step 16, the highly-rated left word extension FEIGENBAUM to the phrase AND+FELDMAN+] looks sufficiently promising that processing now continues in a more depth-first manner along the path FEIGENBAUM+AND+FELDMAN+] through step 25.[7] Processing on the path [+ARE+REDDY does not resume until step 26.

17. KS: PREDICT & VERIFY*
    Stimulus: FEIGENBAUM+AND+FELDMAN+]* (phrase)
    Action: Predict eight words to left. Reject one (DISCUSS). Find two already on blackboard: BY*(80,52:72) and ABOUT(75,48:72). Verify five others: NOT(75,49:82), ED (75,67:72), CITE(70,49:82), QUOTE(70,49:82), CITES(65:49:82).

In steps 18 through 24, alternative word extensions of FEIGENBAUM+AND+FELDMAN+] are explored. As a result of this exploration, the phrase BY+FEIGENBAUM+AND+FELDMAN+] is considered the most credible.

18. KS: CONCAT
    Stimulus: BY* (word), FEIGENBAUM+AND+FELDMAN+]* (phrase)
    Action: Create phrase: BY+FEIGENBAUM+AND+FELDMAN+]* (84,52:225)

19. KS: CONCAT
    Stimulus: ABOUT (word), FEIGENBAUM+AND+FELDMAN+]* (phrase)
    Action: Create phrase: ABOUT+FEIGENBAUM+AND+FELDMAN+] (83,48:225)

---

7 The rating on an hypothesis is only one parameter used by the scheduler to assign priorities to waiting KS instantiations. In particular, the length of an hypothesis is also important. Thus, FEIGENBAUM with a rating of 80 looks better than REDDY with a rating of 85 because it is much longer.

20. KS: PREDICT & VERIFY
    Stimulus: ABOUT+FEIGENBAUM+AND+FELDMAN+] (phrase)
    Action: Predict one word to left: WHAT. Verify it (10,20:49).

21. KS: CONCAT
    Stimulus: CITE (word), FEIGENBAUM+AND+FELDMAN+] (phrase)
    Action: Create phrase: CITE+FEIGENBAUM+AND+FELDMAN+] (83,49:225)

22. KS: PREDICT & VERIFY
    Stimulus: CITE+FEIGENBAUM+AND+FELDMAN+] (phrase)
    Action: Predict four words to left. Reject two of them: BOOKS, PAPERS. Verify THESE (25,28:49), YEAR(20,30:49).

23. KS: PREDICT & VERIFY*
    Stimulus: BY+FEIGENBAUM+AND+FELDMAN+]* (phrase)
    Action: Predict ten words to left. Reject five: ABSTRACTS, ARE, BOOKS, PAPERS, REFERENCED. Find two already on blackboard: ANY*(65,24:49), THESE(25,28:49). Verify three more: ARTICLE(25,9:52), WRITTEN(25,24:52), ARTICLES(10,9:52).

24. KS: CONCAT
    Stimulus: NOT (word), FEIGENBAUM+AND+FELDMAN+]*
    Action: Create phrase: NOT+FEIGENBAUM+AND+FELDMAN+] (83,49:225)

25. KS: CONCAT*
    Stimulus: ANY* (word), BY+FEIGENBAUM+AND+FELDMAN+]* (phrase)
    Action: Create phrase: ANY+BY+FEIGENBAUM+AND+FELDMAN+]* (82,24:225)
    [+ARE+ANY+BY+FEIGENBAUM+AND+FELDMAN+]* (85,0:225) is also created, from [+ARE+ANY and BY+FEIGENBAUM+AND+FELDMAN+].

In step 25, the word ANY is concatenated onto the phrase BY+FEIGENBAUM+AND+FELDMAN+]. However, instead of only creating this new combined phrase, the CONCAT KS also notices that the word ANY is the last word of the phrase [+AND+ANY; this leads the CONCAT KS to merge the two adjacent phrases [+ARE+ANY and BY+FEIGENBAUM+AND+FELDMAN+] into a single enlarged phrase, after first ascertaining that the resulting phrase is grammatically allowed. This merging bypasses the several single-word PREDICT, VERIFY, and CONCAT actions that would be necessary to generate the enlarged hypothesis from either of the two original hypotheses in an incremental fashion. Thus, the recognition process is sped up, not only because the several single-word actions are eliminated, but also because KS actions on competing non-correct hypotheses are avoided since these actions do not appear to the scheduler as attractive as actions on the new, enlarged hypothesis. Such mergings occur in approximately half of the runs on the 1011-word grammar with the small branching factor ("X05"); in grammars with higher branching factors, the merging of phrase hypotheses occurs with even higher frequency.

It has been our experience that, just as a multi-word island is more credible than the individual words that compose it, so a merged phrase hypothesis is more credible than its two constituent phrases. For example, about 80% of the mergings in X05 runs produce correct hypotheses. In more complex grammars, this statistic drops to about 35%, but there are correspondingly more phrase mergings that occur.

The newly-created merged phrase also happens to be a complete sentence; i.e., it has begin- and end-of-utterance markers at as its extreme constituents. Thus, it is a candidate for the interpretation of the utterance.

26. KS: STOP
Stimulus: [+ARE+ANY+BY+FEIGENBAUM+AND+FELDMAN+]* (complete phrase)
Action: Deactivation of several score hypotheses.

STOP responds to the creation of a complete phrase. STOP tests each phrase hypothesis on the blackboard to see whether there is any possibility of extending it to produce a complete phrase that is rated higher than the one just created. It performs this heuristic test by trying to combine the phrase, just based on simple time adjacency constraints, in the best possible way with words already hypothesized. Each phrase that cannot be extended by this process into a word sequence that spans the entire utterance and is better than the newly created complete phrase is discarded. Subsequently, the RPOL KS (whose executions are not shown here) will discard hypotheses as they are created if they also cannot pass the same test.

Of the hypotheses not discarded, extensions to the phrase [+ARE now appear as the most likely candidates to produce new and better complete phrases. This search for better complete phrases results, in steps 27 through 36, in the examination of numerous alternative extensions, each of which is promising.

27. KS: PREDICT & VERIFY
Stimulus: [+ARE+REDDY
Action: Predict three words to right. Verify CITED(60,52:86), OR(30,52:67), AND (25,52:82).

28. KS: CONCAT
Stimulus: [+ARE (phrase), HUGH (word)
Action: Create phrase: [+ARE+HUGH (83,0:39)

29. KS: CONCAT
Stimulus: [+ARE (phrase), YOU (word)
Action: Create phrase: [+ARE+YOU (83,0:39)

30. KS: CONCAT
Stimulus: [+ARE+REDDY (phrase), CITED (word)
Action: Create phrase: [+ARE+REDDY+CITED (80,0:86)

31. KS: PREDICT & VERIFY
Stimulus: [+ARE+REDDY+CITED (phrase)
Action: Predict two words to right. Verify BY(75,83:98), IN(20,86:114).

32. KS: CONCAT
Stimulus: [+ARE+REDDY+CITED (phrase), BY (word)
Action: Create phrase: [+ARE+REDDY+CITED+BY (80,0:98)

33. KS: PREDICT & VERIFY
Stimulus: [+ARE+REDDY+CITED+BY (phrase)
Action: Predict one word to right. Verify ANY(30,105:126).

34. KS: PREDICT & VERIFY
Stimulus: [+ARE+HUGH (phrase)
Action: Predict one word to right. Verify NAGEL(40,42:63).

35. KS: PREDICT & VERIFY
    Stimulus: [+ARE+YOU (phrase)
    Action: Predict three words to right. Reject USUALLY. Verify REGULARLY(25,39:116), ALWAYS(15,39:72).

36. KS: CONCAT
    Stimulus: [+ARE+REDDY (phrase), OR (word)
    Action: Create phrase: [+ARE+REDDY+OR (79,0:67)

37. KS: CONCAT
    Stimulus: [+ARE+REDDY (phrase), AND (word)
    Action: Create phrase: [+ARE+REDDY+AND (78,0:82)

38. KS: STOP
    Stimulus: Stagnation
    Action: Stop search and accept [+ARE+ANY+BY+FEIGENBAUM+AND+FELDMAN+]*.

KS STOP is again executed; this execution is caused by the lack of any KS instantiations that are rated sufficiently high. STOP here makes a decision to terminate the search process and accept the phrase [+ARE+ANY+BY+FEIGENBAUM+AND+FELDMAN+] as the correct interpretation.

39. KS: SEMANT*
    Stimulus: Recognized utterance: [+ARE+ANY+BY+FEIGENBAUM+AND+FELDMAN+]
    Action: SEMANT parses the utterance, using the same grammar, but with semantic routines on some of the non-terminal nodes. The execution of these routines incrementally produces the following structure:

```
F:[ U:( [ ARE ANY BY FEIGENBAUM AND FELDMAN ] )
    N:( $PRUNE!LIST
        S:($PRUNE!LIST!AUTHOR K:(A:(( FEIGENBAUM * FELDMAN)))) )
  ]
```

    "F" denotes the total message. "U" contains the utterance itself. "N" indicates the main type of the utterance (e.g., REQUEST, HELP, etc.), "S" the sub-type. "K" denotes the different attributes associated with the utterance (e.g., "A" is the author and "T" is the topic).
    This structure is passed on to the discourse component, which queries the data base and responds to the speaker.

CONCLUSIONS

The Hearsay-II system has been successful. It came very close to meeting the ARPA performance goals: In September, 1976, the C2 configuration achieved correct semantic interpretation of 90% of a test set of utterances (with 73% of the utterances being recognized word-for-word correctly). This performance was with the highly constrained "X05" grammar over the 1011-word vocabulary. The test set contained twenty-two utterances, averaging seven words each. These utterances were totally new to the system and were run "blind". The processing time averaged 85 mipss (million instructions per second of speech) on a PDP-10 computer. (Subsequently, some trivial implementation modifications reduced the processing costs to about 60 mipss.)

In addition to its successful performance, the structure of the system is interesting. An attempt was made from the start to develop a clean model for the kinds of complex interactions that would be required of the various sources of knowledge. Although the system was modified substantially as experience was gained, it retained its fidelity to that model, indicating its validity. A detailed discussion of the evolution of the architecture with respect to the model can be found in [Lesser & Erman 77]. Several other problem areas have been attacked with organizations strongly influenced by the Hearsay-II structure: image understanding [Prager et al 77], reading comprehension [Rumumelhart 76], protein-crystallographic analysis [Engelmore & Nii 77], signal understanding [Nii & Feigenbaum 78], and complex learning [Soloway 77].

ACKNOWLEDGMENTS

Raj Reddy has provided much of the vision and energy for this work, most of the central ideas in the Hearsay model, and much technical expertise in many of the knowledge sources in the Hearsay-II system. Richard Fennell and Rick Hayes-Roth have been particularly instrumental in formulating and testing the Hearsay-II architecture. All members of the CMU "speech group" have contributed to this work; their substantial efforts are gratefully acknowledged. Lucy Erman and Mark Fox have made helpful suggestions for this paper.

REFERENCES

Abbreviations: ASSP -- Acoustics, Speech, and Signal Processing
CMU -- Computer Science Dept., Carnegie-Mellon Univ., Pittsburgh, Pa.
IJCAI -- International Joint Conference on Artificial Intelligence

CMU Computer Science Speech Group (1977). Summary of the CMU Five-year ARPA effort in speech understanding research. Technical Report, CMU.

Engelmore,R.S. & H.P.Nii (1977). A knowledge-based system for the interpretation of protein x-ray crystallographic data. Technical Report Stan-CS-77-589, Stanford Univ., Stanford, CA.

Erman,L.D. & V.R.Lesser (1975). A multi-level organization for problem solving using many diverse cooperating sources of knowledge. *Proc. 4IJCAI*, Tbilisi, USSR, 483-490.

Erman,L.D. & V.R.Lesser (1978). System engineering techniques for artificial intelligence systems. In A.Hanson and E.Riseman (Eds.), *Computer Vision Systems*, Academic Press, 1978.

Fox,M.S. & D.J.Mostow (1977). Maximal consistent interpretations of errorful data in hierarchically modelled domains. *Proc. 5IJCAI*, Cambridge, Mass., 165-171.

Goldberg,H., R.Reddy, & G.Gill (1977). The ZAPDASH parameters, feature extraction, segmentation, and labeling for speech understanding systems. In [CMU 77], 10-11.

Itakura,F. (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Trans. ASSP*, 23, 67-72.

Hayes-Roth,F., L.D.Erman, M.Fox, & D.J.Mostow (1977). Syntactic processing in Hearsay-II. In [CMU 77], 16-18.

Hayes-Roth,F., G.Gill, and D.J.Mostow (1977). Discourse analysis and task performance in the Hearsay-II speech understanding system. In [CMU 77], 24-28.

Hayes-Roth,F. & V.R.Lesser (1977). Focus of attention in the Hearsay-II system. *Proc. 5IJCAI*, Cambridge, Mass., 27-35.

Hayes-Roth,F., D.J.Mostow, & M.Fox (1978, in press). Understanding speech in the Hearsay-II system. In *Natural Language Communication with Computers*. L.Bloc (Ed.) Springer-Verlag, Berlin.

Lesser,V.R., R.D.Fennell, L.D.Erman, & D.R.Reddy (1975). Organization of the Hearsay-II speech understanding system. *IEEE Trans. ASSP*, 23, 11-23.

Lesser,V.R. & L.D.Erman (1977). A retrospective view of the Hearsay-II architecture. *Proc. 5IJCAI*, Cambridge, Mass., 790-800.

Lesser,V.R., F.Hayes-Roth, M.Birnbaum, & R.Cronk (1977). Selection of word islands in the Hearsay-II speech understanding system. *Proc. IEEE Inter. Conf. ASSP*, Hartford, Conn., 791-794.

Lowerre,B.T. (1976). The Harpy speech recognition system. Technical Report, CMU (Ph.D. Dissertation).

Lowerre,B.T. & R.Reddy (1978). The Harpy speech understanding system. In *Trends in Speech Recognition*, W.A.Lea (Ed.), Prentice-Hall (this book), Chap. 15.

McKeown,D.M. (1977). Word verification in the Hearsay-II speech understanding system. *Proc. 1977 IEEE Inter. Conf. ASSP.*, Hartford, Conn., 795-798.

Mostow,D.J. (1977). A halting condition and related pruning heuristic for combinatorial search. In [CMU 77], 158-166.

Newell,A., J.Barnett, J.Forgie, C.Green, D.Klatt, J.C.R.Licklider, J.Munson, R.Reddy, & W.Woods (1973). *Speech Understanding Systems: Final Report of a Study Group*. North-Holland. (Originally appeared in 1971.)

Nii,H.P. & E.A.Feigenbaum (1978). Rule-based understanding of signals. In D.A.Waterman & F.Hayes-Roth (Eds.) *Pattern-Directed Inference Systems*, Academic Press.

Prager,J., P.Nagin, R.Kohler, A.Hanson, & E.Riseman (1977). Segmentation processes in the VISIONS system. *Proc. 5IJCAI*, Cambridge, Mass., 642-643.

Reddy,D.R., L.D.Erman, & R.B.Neely (1973). A model and a system for machine recognition of speech. *IEEE Trans. Audio and Electroacoustics*, AU-21, 229-238.

Reddy,D.R., L.D.Erman, R.D.Fennell, & R.B.Neely (1973). The Hearsay speech understanding system: An example of the recognition process. *Proc. 3IJCAI*, Stanford, Cal., 185-193.

Reiser,J.F. (1976). SAIL. Stanford Artificial Intelligence Lab., Memo AIM-289.

Rumelhart, D. E. (1976). Toward an interactive model of reading. Technical Report 56, Center for Human Information Processing, Univ. of Cal. at San Diego.

Smith,A.R. (1976). Word hypothesization in the Hearsay-II speech system. *Proc. IEEE Int. Conf. ASSP*, Philadelphia, Pa., 549-552.

Smith,A.R. (1977). Word hypothesization for large-vocabulary speech understanding systems. Technical Report, CMU (Ph.D. Dissertation).

Smith,A.R. & M.R.Sambur (1978). Hypothesizing and verifying words for speech recognition. In *Trends in Speech Recognition*, W.A.Lea (Ed.), Prentice-Hall (this book), Chap. 7.

Soloway,E.M. & E.M.Riseman (1977). Levels of pattern description in learning. *Proc. 5IJCAI*, Cambridge, Mass., 801-811.

A RETROSPECTIVE VIEW OF THE HEARSAY-II ARCHITECTURE[1]

Victor R. Lesser and Lee D. Erman

ABSTRACT

The Hearsay model has been presented as a paradigm for attacking errorful knowledge-intensive problems requiring multiple, cooperating knowledge sources. The Hearsay-II architecture is the latest attempt to explore the model. This paper describes experiences gained while successfully applying this architecture to the problem of speech understanding. The major conclusions are:

1. The paradigm of viewing problem solving in terms of hypothesize-and-test actions distributed among distinct representations of the problem has been shown to be computationally feasible.
2. A global working memory (the "blackboard"), in which the distinct representations are integrated in a uniform manner, has made it convenient to construct and integrate the individual sources of knowledge needed for the problem solution.
3. The use of a uniform data-directed structure for controlling knowledge-source activity has made the system easy to understand and modify.
4. A solution has been demonstrated to the problem of focus-of-attention in this type of control environment. This solution does not need to be modified when the sources of knowledge in the system are changed.

INTRODUCTION

The Hearsay model [Red73Mo] has been developed for problem-solving in domains which must use large amounts of diverse, errorful, and incomplete knowledge in order to search in a large space.[2] The *Hearsay-I* architecture and system [Red73Hx and Erm74En] represented a first (and successful) attempt to apply that model to the problem of understanding connected speech in specialized task domains. In this first application, the size of the vocabulary (less than 100 words) and complexity of the grammar were very limited.

Experiences with Hearsay-I led to the more generalized *Hearsay-II* architecture [Les75Or and Erm75Mu] in order to handle more difficult problems (e.g., larger vocabularies and less-constrained grammars). The first configuration of knowledge sources (KSs) for Hearsay-II -- configuration C1 -- was complete in January, 1976 [CMU76W4]. This implementation had poor performance (e.g., 10% sentences correct in 85 MIPSS (million instructions per second of speech) on a 250-word vocabulary). Experience with this configuration has led to a substantially different set of KSs -- configuration C2 [CMU77Su]. This configuration performs substantially better (e.g., 85% correct in 60 MIPSS on a 1,000-word vocabulary).

---

1 This paper is slightly modified from its original IJCAI-77 form.

2 Other approaches for solving this class of problem include production systems, frames [Min74Fr], heterarchical structures [Wal77Ov and Woo76Fl], relaxation techniques [Bar76MS and Ros76Sc], Planner [Hew72De], QA4 [Rul73Qa], and the Locus model [Low76Ha and Rub77Lo].

The Hearsay-II system, with the second configuration, has been successful: it comes close to the original performance goals set out in 1971 to be met by the end of 1976 for the ARPA speech understanding effort [New73Sp] and does so with a system organization that is of interest because of the potential for its application to other problem areas. Several other problems have been attacked with organizations strongly influenced by the Hearsay-II structure: image understanding [Pra77Se], reading comprehension [Rum76To], protein-crystallographic analysis [Eng77Kn], signal understanding [Nii77Ru], and complex learning [Sol77Kn].

This paper is divided into two major parts. The first part presents an overview of the Hearsay model, the Hearsay-II architecture, which is a further specification of this model, and the two KS configurations. (More detailed descriptions of these configurations are contained in the appendix.) The second part of the paper discusses the implication of these experiences for the Hearsay model and the Hearsay-II architecture. In particular, those aspects of the architecture are identified that have contributed most strongly to the success of the system, as well as those parts that need the most future work.[3] This discussion is structured around two themes -- the multi-level global data base (blackboard) for KS cooperation, and the asynchronous, data-directed control structure for KS activation.[4]

OVERVIEW OF THE HEARSAY MODEL

A number of characteristics of the problem drive the Hearsay model:

1. Large search space.
2. Diverse sources of knowledge. Many of the KSs are large; some have large internal search problems of their own.
3. Error and variability. These are characteristics of both the input data (the acoustic signal) and the processing of knowledge sources.
4. Experimental approach needed for system development. This implies the need for iterating the system and running over large amounts of data.
5. Performance requirement -- accuracy and speed. This is true of any practical solution to the problem as well as during development (because of the experimental nature).

The basic notions of the Hearsay model [Red73Mo] were developed in response to the requirements just stated:

1. The KSs are kept separate, independent, and anonymous. This separation is felt to be a decomposition which is natural and also can help make the combinatoric problems more tractable. For development purposes, the separation should help with system modifications (especially adding and modifying KSs) and evaluation.

---

3 The fact that certain parts of the implementation need further work does not necessarily indicate deficiencies with the basic Hearsay model, but rather points out inadequacies in the Hearsay-II implementation of the model. It is to the model's credit that even though some of its more sophisticated capabilities are not implemented effectively, it still provides an appropriate framework for the successful solution of a complex task. Thus, one of the intents of this paper is to define some of the major design goals for the next iteration in the implementation of the Hearsay model.

4 While this paper discusses the means of organizing the knowledge and applying it to the problem, it does not describe in detail nor quantify the knowledge in the system. At least as much work has been expended on specifying and debugging the knowledge in the system as on building and refining the structure to hold and apply that knowledge.

2. A global data structure -- the *blackboard* -- is the means of communication and interaction of KSs. This provides an hypothesize-and-test means of interaction. Each KS accesses and modifies the blackboard in a uniform way.
3. A KS responds to changes to the blackboard which it is concerned with; it applies its knowledge within the context of such a change. This implies data-directed activation of KSs.

OVERVIEW OF THE HEARSAY-II ARCHITECTURE

The Hearsay-II architecture is one framework for implementing the Hearsay model. In this section, a very brief overview of that architecture is given. More details are described in [Les75Or and Erm75Mu].

The Blackboard

The blackboard is partitioned into distinct information *levels*; each level is used to hold a different representation of the problem space. (Examples of levels are "phrase", "word", "syllable", and "segment".) The decomposition of the problem space into levels is a natural parallel to the decomposition of the knowledge into separate KSs. For most KSs, the KS needs to deal with only a few (usually two) levels to apply its knowledge. Its interface to the rest of the system is in units and concepts that are natural to it.

The sequence of levels forms a loose hierarchical structure in which the elements at each level can be described approximately as abstractions of elements at the next lower level. The possible hypotheses at a level form a problem space for KSs operating at that level. A partial solution (i.e., a group of hypotheses) at one level can be used to constrain the search at an adjacent level. For example, consider a KS which can predict and rate words based on acoustic information and another KS which knows about the grammar of the language. The first KS can generate a set of candidate word hypotheses. The second KS can use these hypotheses to generate phrase hypotheses which can be used, in turn, to predict words likely to precede or follow. These predictions can now constrain the search for the first KS.

Associated with each level is a set of primitive elements appropriate for representing the problem at that level; e.g., the elements at the word level are the words of the vocabulary to be recognized. The major units on the blackboard are *hypotheses*. An hypothesis is an interpretation of a portion of the spoken utterance at a particular level. E.g., an hypothesis might represent the assertion that the word "GIVE" was spoken at the beginning of the utterance. Each hypothesis at a given level is labeled as being a particular element of the set of primitive elements at that level.

Each hypothesis, no matter what its level, has a uniform attribute-value structure. Some attributes (and values) are required of all hypotheses and others are optional, as needed. Included among the required attributes of an hypothesis are its level (e.g., word), its element name (e.g., "GIVE"), and an estimate of its time coordinates within the spoken utterance (which can include notions of "fuzziness" of estimate). The level and time attributes place a two-dimensional structure on hypotheses which partitions the blackboard and can be used for addressing hypotheses. Note that two or more hypotheses at the same level with significantly overlapping times are *competitors*; i.e., they represent competing interpretations of a portion of the utterance.

Other attributes of an hypothesis include information about its *structural relationships* with other hypotheses (forming an AND/OR graph), *validity ratings* (i.e., estimates by KSs of the "truth" of the hypothesis), and *processing state*. The processing state attributes are summaries and classifications of the other attributes. E.g., the values of the rating attributes are summarized by the "rating state" attribute that takes a value from the set "Unrated", "Neutral", "Verified", "Guaranteed", or "Rejected". New attributes can be created by any KS and may be used for passing arbitrary information about an hypothesis between instantiations of the same or different KSs.

A KS can create new hypotheses, specifying values for attributes of the new hypothesis. Given the "name" of an hypothesis, a KS can examine or modify attributes of that hypothesis. In addition, sets of hypotheses may be retrieved associatively, based on the values of their attributes (e.g., all hypotheses at the syllable level whose durations are greater than 250 msec.). The hypothesis structure is uniform across all levels in the blackboard. Thus, the form of access and modification to hypotheses by KSs can also be uniform and is accomplished by calling kernel procedures; the set of these procedures comprises the *blackboard handler*.

In addition to the information in each hypothesis which can be accessed by KSs, *auxiliary state information* is maintained by the blackboard handler in specialized data structures. Examples of this information are (1) a representation of hypotheses at each level arranged for efficient associative retrieval by time and (2) the name of the highest-rated hypothesis in each time area. These auxiliary structures are updated by the blackboard handler automatically as KSs make changes to the blackboard.

Structure of Knowledge-Sources

Each KS has two major components: a *precondition* and an *action*. The purpose of the precondition is to find a subset of hypotheses that are appropriate for action by the KS and to invoke the KS on that subset; the subset is called the *stimulus frame* of the KS instantiation. For example, the precondition of the KS that generates word hypotheses based on syllables looks for new syllable hypotheses. When invoking the KS, the precondition provides the system scheduler with, in addition to the stimulus frame, a stylized description of the likely action that the KS instantiation will perform (if and when it is allowed to execute); this estimate of action is called the *response frame*. For example, a response frame for the syllable-based word hypothesizer (MOW) indicates that the action will be to generate hypotheses at the word level and in a time area that includes at least that of the stimulus frame. The action part of a KS is a program (written in SAIL [Rei76SA]) for applying the knowledge to the stimulus frame and making appropriate changes to the blackboard. In general, the changes made will serve to trigger more KS activations.

To keep from having to fire the precondition continuously to search the blackboard, each precondition declares to the blackboard handler in a non-procedural way the primitive kinds of blackboard changes in which it is interested. Each precondition is triggered only when such primitive changes occur (and is then given pointers to all of them). This changes a polling action into an interrupt-driven one and is more efficient, especially as the number of preconditions gets large. After being triggered (and when scheduled for execution), the precondition (also a SAIL procedure) can do arbitrary searching of the blackboard for hypothesis configurations of interest to its KS.

Several KSs may be grouped together into *modules*. The KSs within a module may share code and long-term built-in data. A discussion of the module construct, including its implications for KS independence, is given below in the section on "KS Independence".

Scheduling

Whenever a precondition is executed, it checks all blackboard events in which it is interested that have occurred since the last time it executed. For example, a "new hypothesis" to a precondition is any hypothesis which was created between the last time the precondition executed and its current execution. Thus, a precondition may be thought of as executing, then "sleeping" for a time while retaining state, then waking (executing again) and being able to find all new events of interest to it.

However, whenever a *KS* executes, it uses the stimulus frame specific to that invocation. Each KS execution goes to completion; that is, the KS cannot put itself to "sleep", waiting for some other event (on the blackboard) to occur.

At any point, there are, in general, a number of pending tasks to execute -- both invoked KSs and triggered preconditions. (In practice, the number of pending tasks often exceeds 200.) A *scheduler* in the kernel [Hay77Fo] calculates a priority for each waiting task and selects for execution the task with the highest priority. The priority calculation attempts to estimate the usefulness of the action in fulfilling the overall system goal of recognizing the utterance. This estimation is based on the specific stimulus and response frames of the actions and on overall blackboard state information, which includes such notions as the best hypotheses in each time area in the utterance, and how much time has elapsed since the current best hypothesis was generated. The priority of a KS is recalculated if the validity of its stimulus frame is changed or the auxiliary state pertinent to evaluating the significance of the response frame is modified.

Some KSs are not directly involved in hypothesizing and testing partial solutions; instead, these control the search by influencing the activation of other KSs. These *policy* KSs can be used to impose global search strategies on the basic priority scheduling mechanism.

THE CONFIGURATIONS

Following are brief overviews of configurations C1 and C2, to provide a basis for subsequent discussion. The appendix contains more detailed descriptions of the KSs, as well as pointers to published papers.

Figure 1 gives a schematic of configuration C1 as it was operational in January, 1976. The levels are indicated by solid horizontal lines and are labeled at the left. KSs are indicated by vertical arcs with the circled end indicating the level where its stimulus frame is and the pointed end indicating the level of its response frame. The name of a KS is connected to its arc by a dashed horizontal line. As segment hypotheses were generated from the acoustic data (SEG), they might be combined to form larger segment hypotheses (CSEG). Phone hypotheses were created, based on one or more contiguous segments (PSYN). Syllables were predicted from the phones (POM) and words from the syllables (MOW). Phrase hypotheses were constructed from contiguous word or phrase hypotheses which were syntactically consistent (RECOG). Other KSs (PREDICT, RESPELL, and POSTDICT) accomplished various

syntactic extension and prediction functions at the phrase and word levels. Verification of predicted words was carried out by expanding the words into their expected syllables (WOM), expanding the syllables into expected phonemes (MOS), and matching the sequences of expected phonemes with the recognized phones (TIME and SEARCH). Changes of ratings of hypotheses were propagated to structurally connected hypotheses (RPOL). The FOCUS policy KS controlled the search by setting priorities for various kinds of KS actions.

Figure 2 gives a schematic of configuration C2 as it was operational in September, 1976. First, all segment hypotheses are generated from the parametric representation of the acoustic signal (SEG). Next, syllables are predicted from the segments (POM). Then, words are predicted from the syllables (MOW); the most likely words in each time interval placed on the blackboard (WORD-CTL). Next, a heuristic word-sequence hypothesizer (WORD-SEQ) attempts to identify the most probable sequences of word hypotheses (consisting of successive language-adjacent word pairs). Because this KS exploits statistical methods to improve credibility, the initial word sequence hypotheses are much more accurate than are hypotheses based on single words. Subsequently, KSs are invoked to attempt to parse the hypothesized word sequences to determine if they are grammatical (PARSE), to predict possible time-adjacent grammatical word extensions (PREDICT), to hypothesize and verify new words satisfying these predictions (VERIFY), to concatenate grammatical and time-adjacent word sequences (CONCAT), to propagate ratings (RPOL), to reject phrases and to determine when the serach should be terminated (STOP), and to generate new word sequence hypotheses (WORD-SEQ-CTL).

The major system-related differences between these configurations[5] are listed here; they will be discussed individually throughout the paper.

1. C1 has asynchronous processing throughout. C2 has an initial pass of sequential, bottom-up processing to the word level; i.e., all segments are created, then all syllables, then a selection of words.
2. C1 used the blackboard extensively for intra-KS state-saving between instantiations of a KS (e.g., SEARCH and RECOG-PREDICT-RESPELL-POSTDICT). In C2, this was greatly reduced, with KSs doing more computation internally and in larger units (e.g., VERIFY and PARSE-PREDICT-CONCAT).
3. C2 generated simpler hypothesis networks than those in C1. For example, SEARCH and TIME built complex structures to represent verifications of words; VERIFY builds very simple ones for the same purpose.

EXPERIENCES WITH HEARSAY-II

This section addresses the following questions: How well did the Hearsay-II system meet its original design goals and were these goals appropriate for problem solving in the speech understanding domain (and more generally in errorful domains which require extensive search)? This discussion is based on approximately three years of experience with the Hearsay-II architecture, including numerous iterations of both the system architecture and KS

5 Though we are here concerned with systems issues, it is worth pointing out that WORD-SEQ is a novel KS which significantly contributes to the success of C2. It limits the search space by providing large hypotheses which act as islands of reliability and bases for further search. This KS uses *approximate* syntactic knowledge to examine efficiently many alternative sequences of low-reliability word hypotheses and generate a small number of more reliable phrase hypotheses.

configurations.[6] These questions will be discussed in the context of two major aspects of the Hearsay-II architecture: the blackboard global data base, and KS interaction and control.

Blackboard Data Base

There are two major design themes reflected in the structure of the blackboard. The first theme is the avoidance of expensive and complicated backtracking control structures by the representation of alternative, distributed hypotheses in an integrated multi-level manner. The second design theme is the representation of all information levels with a high-level, uniform structure, in order to allow all KSs to contribute their information to the blackboard in an identical and anonymous manner.

Distributed Representation

It was hoped that the first design theme would (a) avoid the redundant calculation of previously-generated results and (b) allow KSs to apply their knowledge selectively to places in the blackboard where further processing would resolve contradictory evidence supporting likely, alternative hypotheses.[7]

The ability to save partial results on the blackboard in an integrated manner, in terms of hypothesis sub-networks, has been a very positive characteristic of the architecture; it avoids a significant amount of unnecessary recalculation of results previously generated.[8] This was especially true for KSs operating at the word and phrase levels. This was also true for KSs in the C1 configuration operating at lower information levels, for example, the TIME and SEARCH KSs. However, later versions of these KSs (e.g., VERIFY in C2), for reasons of efficiency (to be discussed later), do not save partial results on the blackboard.

The use of an integrated representation as a way of efficiently resolving competition among KSs wanting to work on the same hypotheses has not been exploited, nor has the ability to bring to bear specialized knowledge dynamically to resolve the conflict among competing, alternative hypotheses (for example, a specialized KS to resolve ambiguity between two word hypotheses that are very close acoustically -- e.g., "sit" and "split"). In addition, the ability given by the integrated representation to re-evaluate automatically (i.e., without KS intervention) an hypothesis' credibility when its supporting environment is modified is not exploited in the C2 configuration (although it was C1). In the C2 configuration, hypothesis credibility is never modified in an explicit sense; rather, new and different hypotheses are created. A side effect of this approach is that hypotheses are never deleted from the blackboard.

---

6 The emphasis on the two configurations as fixed points can be misleading; rather than appearing full-grown, the configurations evolved over time, with numerous iterations required first to develop C1 and then C2 from C1.

7 Hayes-Roth [Hay77Ro], in discussing how to evaluate the potential usefulness of a KS action, introduces the concept of *diagnosticity* as an important component in a KS priority function. Diagnosticity is a measure of how much contradictory evidence could potentially be resolved by a particular KS action.

8 The usual manner of accomplishing this is having each KS, as it is about to create a new hypothesis, first check that a hypothesis does not already exist which is sufficiently similar to the one it is about to create.

One explanation for the lack of full use of the integrated, multi-level representation of hypotheses could be just that the particular task domain of speech understanding does not need these capabilities. However, it is our feeling that there are fundamental weaknesses in the Hearsay-II representation of an integrated, multi-level hypothesis; these weaknesses (to be discussed below) make it difficult, both in terms of execution time and programming complexity, to perform the desired analyses of the hypothesis structure and its surrounding environment. This type of analysis is the key to the effective use of the sophisticated processing capabilities that are possible within the framework of the Hearsay model.

Hypothesis Network Structure

A major problem in using the blackboard is that one cannot operate on a network (in its simplest form, a tree) of interconnected hypotheses as a composite unit. There is a basic confusion in Hearsay-II's implementation of hypothesis networks between (a) the hypothesis at the top of the tree (the highest level of interpretation) and (b) the whole tree; the state information associated with an hypothesis is very local and does not adequately characterize the state(s) of the hypothesis network(s) connected to it. In order to operate effectively in a distributed manner on interconnected multi-level hypothesis networks, the state information associated with an individual hypothesis must allow a KS to analyze quickly the local environment of an hypothesis and, more importantly, the role that the hypothesis plays in the larger context of the hypothesis networks it is part of. One of the consequences of this deficiency is the difficulty encountered in making appropriate scheduling decisions because the more global import of a potential KS action cannot be determined easily.[9]

For example, in configuration C1, an hypothesis at the phrase level was constructed out of hypotheses at the phrase, word, syllable, surface-phoneme, phone, and segment levels. Because of the asynchronous nature of processing, a phrase hypothesis could be supported by word hypotheses in different stages of verification -- some might be fully verified, others only partially verified, or some totally unverified. Possible KS actions waiting to work on this hypothesis network could be a separate verification of each unverified word, an attempt to extend the phrase in either the right or left direction, a search for co-articulation effects among different word pairs, or a full verification of a partially verified word. These actions represent processing at different information levels. Given the existing hypothesis interconnection primitives, there is no way to determine easily that all these actions relate to the same hypothesis network, nor what import each action could potentially have in judging the credibility of the entire network.

Another symptom of this problem is the inability to express, except in a very limited way, what type of processing has already been applied to an hypothesis network and what further processing could possibly be applied. This inability again impacts the scheduler because it makes it difficult to schedule "competing" KSs (i.e., KSs which could work on the

9 It is expensive to trace through an hypothesis network to determine the global import of a potential KS action. But this cost is not unreasonable relative to the total system execution time for a configuration which contains KSs that perform moderately large amounts of internal computation. However, the major computational expense comes in dynamically updating the global import of a pending KS action as modifications are made to the blackboard since there are a large number of these modifications: it is necessary both to find which waiting KS instantiations have priorities that are affected by the modification and then to recalculate the priorities for those affected.

same or different parts of a specific hypothesis network) appropriately. Because of these difficulties there has been, in later KS configurations, only a very limited (and simply represented and analyzable) form of KS competition.

Another aspect of the inadequate network structure is that the primitives for specifying structural relationships between hypotheses require many intermediate levels to represent certain types of connectivity patterns. This need for many intermediate levels is expensive in in storage space and, more importantly, time; it requires a great deal of network searching through the connection structure to analyze the relationship of an hypothesis to its immediate surrounding environment. These intermediate levels represent a level of detail which is unnecessary for some types of KS analysis and which interfere with these analyses by making them unwarrantedly complex. Once it has been constructed, it is impossible to bypass this level of detail in situations in which it is not pertinent. For example, an information level may contain many intermediate sublevels built out of the connection primitives; a KS using information at this level may want only to examine those hypotheses which are the highest sublevel in each time area. This type of operation, given the current blackboard retrieval primitives, requires the examination of all hypotheses in a specified time area. Another complication of not being able to hide these intermediate levels is that a KS in some cases has to know the exact structure of the intermediate levels used by another KS in order to be able to skip over them, thus making the KSs less independent.

In summary, the experience to date on the distributed representation approach indicates that the implementations of this concept explored so far are neither general nor efficient enough in two major interrelated aspects -- how hypotheses can be combined into a network and how the state information associated with an individual hypothesis reflects the hypothesis networks connected to it. To elaborate further, what is missing from the blackboard structure is a way of viewing the shared network structure from a different perspective. This perspective should permit the particular path through the network that defines a specific composite hypothesis to be both viewed in isolation from other paths that are intertwined with it, and also in a way that eliminates superfluous sub-structure. From this type of perspective, the importance of potential KS actions could be judged efficiently and related to the history of previous processing.[10]

Uniform Blackboard Structure

Let us now examine the second major design theme used to structure the blackboard: a uniform structure at all information levels. From a programming point of view, both in terms of KS writers and system implementors, the uniform structure of the blackboard has been a good design choice. By having a uniform structure, a variety of standard blackboard creation, accessing, display, analysis, and debugging functions could be developed that are usable by all KSs. These standard functions, some of which are quite complex, make it convenient for a KS writer to interface his knowledge source with the system. The ease with which this interfacing could be accomplished is exemplified by the fact that, in a period of six months, configuration C2, which is almost entirely new relative to C1, was developed and debugged. Because of this uniform structure of hypotheses and their connections, it is often possible for a KS to be recoded so that it generates a different local hypothesis structure without requiring the recoding of other KSs in the system; this is true because a KS can probe the

10 A possible approach for implementing this different type of perspective is discussed in work by Hendrix [Hen75Ex] on partitioned semantic networks.

blackboard with sophisticated built-in retrieval operations which, in many cases, shield the KS from changes made by other KSs. For example, there is the *structural-adjacency* blackboard primitive which, given a hypothesis, finds all hypotheses at a particular information level that are immediately adjacent to the given hypothesis based on the AND/OR connection structure among hypotheses.

The uniformity of the attribute structure of hypotheses also makes it possible to monitor efficiently for blackboard changes which are to trigger preconditions. Each precondition needs only to declare to the blackboard handler the names of the attributes at each level in which it is interested. When an attribute is changed, the blackboard handler then triggers all preconditions interested in it.

The uniform blackboard structure, though efficiently implemented, is not appropriate as a scratchpad for the internal computations of a KS. This type of use of the blackboard is often inappropriate because its uniform, general structure does not come completely free in the storage requirements for an hypothesis and the cost of creation and access; most internal computations of a KS do not need this generality. An example of a misuse of the blackboard was the case of the syntax analyzer knowledge source, SASS [Hay77Un]. In early versions of this KS, the blackboard was used to hold the partial parse trees developed in attempting to parse a language fragment; current versions of this KS, which use a tailored, internal data structure for parsing, are two orders of magnitude faster than the original blackboard-based version of this KS. This case history seems to confirm the notion that there are advantages to specialization of structures: one for KS interaction (i.e., the blackboard), and separate ones for each KS.

The blackboard has also proven to be useful as a data base for the scheduler [Hay77Fo]. Because of the uniform hypothesis structure, instantiations of KSs can specify scheduling information in a uniform way (as stimulus and response frames), allowing new KSs to be introduced without having to modify the scheduler. The representation of alternative hypotheses in an integrated, uniform fashion also makes it possible to compare directly the pending KS instantiations to determine which will likely contribute most to further progress; the scheduler 1) can determine those areas on the blackboard that most need further work and locate the pending KS instantiations that are relevent to those areas and 2) estimate the amount that a KS instantiation will improve the quality of hypotheses in the area of its action.

Long-Term Information Structures

Associated with each information level of the blackboard, there is, as previously discussed, a set of primitive elements that are used to label hypotheses at that level. The kernel interface provides facilities for creating, accessing, and displaying these labels. In addition, arbitrary data structures can be associated with each label. These structures, for example at the word information level, can be simple, such as the average expected duration of each word, or complex, such as a network which specifies alternative syllabic spellings for each word. In the complex case, this structure often is used to relate labels at one information level with labels at another; this relationship is used by a KS which operates between different levels (e.g., in the example given here, WOM in configuration C1). These data structures related to labels constitute much of the long-term (built-in) KS-defined information structures of the system and often represent most of the problem-specific knowledge in a KS.

Each KS (or group of KSs) defines whatever ad hoc structure seems appropriate for the particular kind of information to be represented. There has been no attempt to define a uniform set of kernel interfaces for creating and accessing these long-term data structures, nor a set of relationships (connection primitives) for relating labels at different levels. However, it seems possible to attempt to define a small number of representations within the kernel; these structures would mimic the hierarchical structure of the blackboard. (Hanson and Riseman in their work on image understanding have a system architecture [Pra77Se] very similar to the blackboard and have included a complementary long-term memory structure.)

The major drawback of not having a predefined long-term memory is that if KSs want to share this information they have to agree among themselves upon a specific structure, thus violating independence considerations. In addition, uniform structures could make KSs easier to understand, develop, and analyze.

On the other hand, these long-term structures must be highly optimized because of their large size and the high frequency with which they are accessed.[11] The approach taken of tailoring these structures to the particular KS(s) using them allowed for efficient implementations in terms of both time and space. It is also possible that explicit tailoring has led to KSs which are easier to understand than if they were forced to fit their requirements into a uniform structure.

Thus, there are still open questions about the desirability of providing uniform structures for representing the knowledge in KSs; hopefully, future implementations will explore these possibilities.

Conclusions About Blackboard Usage

In trying to draw some conclusions about our experiences with the use of the blackboard, the main issue that constantly comes up is time and space efficiency. In errorful task domains, such as speech understanding, a large number of alternative interpretations of the data must be examined and analyzed. The blackboard concept is effective in the Hearsay-II implementation to the degree that it allows this search to be efficient. Analysis of the C1 configuration indicated that certain types of KS processing on the blackboard were not efficient. Reimplementation of the KSs in order to eliminate those types of processing resulted in the C2 configuration. The major uses of the blackboard in the C2 configuration are:

1. A storage area for high-level intermediate results generated by the search. This storage area avoids the unnecessary recalculation of these results if they are encountered on future search paths.
2. A communication area for KSs, with strong and simplified assumptions by a KS of what structures can be generated by other KSs.
3. A data base for the scheduler.
4. A common display, debugging, and performance evaluation area.

---

11 For example, the description of the grammar used by the KSs within the SASS module in configuration C2 is a network of 3100 nodes. Each node has about seven pointers to other nodes, plus several pieces of auxiliary information. A typical KS action, e.g., parsing a four-word phrase, might make 100 to 5,000 node accesses.

Knowledge-Source Interaction and Control

The asynchronous, data-directed control structure used in Hearsay-II was designed to permit:

1. The quick refocusing of attention to appropriate hypotheses in the blackboard.
2. The flexible reconfiguration of the system with different sets of independent (and possibly competing) KSs, and different global control strategies.
3. The exploration of parallel processing.

This section will examine each of these requirements along two dimensions: Were the capabilities embodied in the requirement important to the project, and how well did the control structure (in terms of time, space, and ease of representation) implement these capabilities?

Appropriateness of a Data-Directed Control Structure

The first requirement, quick refocussing, was based on the following model for processing in the speech domain. Processing can be organized in terms of the incremental additions of small units of information to a limited number of alternative hypotheses. The limited number of alternatives derives from the view that there are islands of reliability in the acoustic data that can be used to anchor the search. Each small increment of information should help to verify, refute, or augment (expand) an hypothesis. A KS action, though performed in a local context, could also have the side effect of contributing information useful in the evaluation of alternative hypotheses (i.e., in other contexts). Thus, after each *incremental addition of information* (through the execution of a KS), it is necessary to re-examine the set of potential actions that now can be activated and determine which of these will most likely resolve ambiguity. An asynchronous, data-directed architecture makes it convenient to implement such a processing strategy by permitting KS action to be directed by the data: it delays the application of knowledge until there is enough information for a meaningful result (decision), and it re-applies the knowledge when, at a later time, additional information is generated that bears on the original decision.

In those parts of the blackboard where processing followed this model, the data-directed control structure was very effective. However, at lower levels of speech processing (i.e., segmentation and labeling, syllable hypothesis generation based on segments, and word spotting based on syllables), this model was found to be inappropriate because there is not enough reliability in credibility scores of hypotheses to form hypothesis islands that can reliably anchor the search. Thus, processing at these levels cannot be selective (depth-first), and instead requires a complete scan (breadth-first), for which asynchronous control has no advantages (and considerable costs).

A major change in going from configuration C1 to C2 was making the lower levels of processing more sequential and bottom-up. Not until the word level is reached do hypothesis credibility scores have enough reliability to justify the more complex processing required of an asynchronous, data-directed control structure. The presence of these islands of reliability is in itself not a sufficient condition for the use of this sophisticated control structure. What is additionally required is that there is either a significant cost to evaluate each alternative or a large number of alternatives (combinatoric explosion in the search space); only then is the overhead involved in implementing a data-directed control structure worthwhile.

In addition to the control overhead, an asynchronous control structure requires a more

complex internal structure for a KS. This complexity arises because, as new information is asynchronously generated, a KS must have the additional logic to determine whether this new information allows it to make a decision it could not previously make or whether this information contradicts a previous decision. In the latter case, it must modify the previous decision, which may involve modifying decisions made as a consequence of the original one. Where processing involves a complex hypothesis network structure with much detailed structure, the nature of asynchronous processing in response to a change at the detailed level is costly, both in terms of processing time and complexity of the KS, and should be avoided unless the compensatory benefits are large. As previously mentioned, the inadequacies in the blackboard structure which make it difficult to skip over detailed structure exacerbate these problems. (The SEARCH KS in configuration C1 is an example of a KS working asynchronously at a detailed level. Although the acoustic-phonetic knowledge applied by SEARCH was represented by a relatively simple data structure within the KS, the code necessary for examining and incrementally building large, integrated, and competing AND/OR structures on the blackboard was very complex and the number of KS executions needed to verify a word was large -- on the order of ten to one hundred. In C2, the function of word verification was replaced by the VERIFY KS -- here, verifying a word is an atomic act (as far as other KS actions are concerned) and is carried out using tailored structures internal to the KS. Each execution of VERIFY forced a recalculation of the detailed structure, rather than sharing such structures across executions.)

Overhead Costs of Data-Directed Control

The overhead cost of implementing an asynchronous data-directed control structure for computation of medium level granularity (i.e., a KS action which involves greater than 1/10 second of internal computation) is not significant. The major cost involves monitoring each modify operation to the blackboard to determine whether any preconditions are interested in being notified of this specific change. This cost of monitoring and notification makes a modify operation 12 times as expensive as a read operation. However, in the C2 configuration there are 29 times as many reads as modify operations, thus making this aspect of implementing a data-directed control only 4% of the total cost of a run.

Another cost associated with implementing this type of control structure involves maintaining a scheduler queue of waiting KS instantiations and performing priority calculations to decide which instantiation to run next. However, these focus of control calculations, possibly expressed in a different way, are necessary in any problem-solving system that involves a dynamic search. The more general implementation of these calculations in the context of an asynchronous control structure does not appear to generate significantly more system overhead than a specialized implementation of them in a system with more explicit control structure. The cost of maintaining and updating the scheduler queues and calculating the priorities was about 5% to 7% of a total run.

Further costs involved in implementing this type of asynchronous control structure arise because of the delay between the invocation of a KS and its execution. The KS must, in general, contain code that revalidates its invocation context before beginning execution. However, by making some assumptions about the type of processing other KSs could effect at particular information levels, there was in practice very little need for context revalidation. KSs did not in general interact by modifying previously-made assumptions and detailed structures constructed by other KSs, but rather through the incremental addition of new hypotheses to existing structures or the verification of previously unverified hypotheses.

KS Independence

As indicated above, complete independence among KSs was not accomplished. However, information about the processing characteristics of other KSs is generally very restricted, and relates only to KSs which share either dynamic information on the blackboard or long-term static information. To facilitate such data sharing, the concept of a *module* was introduced into the architecture. A module contains a set of preconditions and KSs which share common structures and related accessing procedures. The KSs contained in a module generally operate at the same or adjacent information levels and thus also share specialized accessing and display routines for these information levels of the blackboard. A module usually represents the code of one KS programmer and typically contains one to four KSs and one to four preconditions. The clustering of KSs by their long-term information structures turned out to be a convenient decomposition for separably instantiable but related activity. The KS module is the atomic unit which is the basic building block for different KS configurations.[12]

How important is the property of independence of KSs? For the two configurations discussed here, the KS modules are not completely independent. However, during the lifetime of the project, which involved numerous iterations of KSs, there has been very little difficulty encountered by this lack of complete independence (i.e., the "subroutine interaction problem" did not haunt us). It has been possible to configure systems with subsets of KS modules (e.g., a "top-end" system that deals only with word and phrase hypotheses or a "bottom-end" system which deals only at and below the word level) without modifications to the modules involved.

The reason for having little difficulty with the subroutine interaction problem can be traced to the data-directed activation of KSs. In general, interaction among KSs is accomplished by having a KS modify the attribute structure of an hypothesis in a way which causes some other KS(s) to be activated and attend to that hypothesis. In order for KSs to communicate information which is not representable using the standard, kernel-supplied attributes, the communicating KSs need only agree on the name of a new attribute and the form of its value; this new attribute can then be used to pass the information. Thus, it is not necessary for a KS to know the names of the other KSs involved. Individual KSs which create, are activated by, or use this information may be added to or deleted from the system without requiring modifications to the other KSs.

A KS as a Hypothesis Generator

There are two major reasons, in addition to the one already discussed about context validation, why total independence was not achieved; both of these relate to a KS as a generator of hypotheses. The first reason concerns the control of the number of hypotheses a KS should initially generate and the reinvocation of it to generate additional, alternative hypotheses. The parameters associated with hypothesis generation should be set by a policy KS which has a more global view of the current state of the recognition process. The need then arises for a mechanism by which a policy KS can transmit its desires, in an anonymous and independent manner, to the appropriate KS.

It was hoped initially that these "processing goals" could be specified in terms of the

12 Each module is implemented as a separately compiled body of code. A configuration is specified at load time by selecting the desired modules. Additionally, any KS or precondition can be *inhibited* at run-time, effectively excising it from the system.

basic hypothesize-and-test paradigm (i.e., by having the policy KS create the appropriate type of hypotheses which would in turn trigger the desired activity). However, "asking for something to be done" cannot always be specified conveniently in this way nor in an anonymous manner. For example, if there is a need for more word hypotheses to be generated in a particular time area, the action of creating a new hypothesis at the phrase level which will then be expanded at the word level does not precisely capture the desired activity, nor does the somewhat clumsy approach of modifying some attribute of the lower level data (e.g., the syllable level) to force a KS to reprocess this data so as to accomplish the desired activity. Note in this example, that by trying to force the concept of processing goal into the hypothesize-and-test paradigm, the policy KS must know the type of input stimulus that will trigger a KS to produce the desired results, thus violating the independence among modules. In addition, a KS which is designed to do hypothesizing-and-testing does not necessarily produce a response that will precisely match the desired processing goal. Due to these difficulties of directly embedding goal processing control in the hypothesize-and-test paradigm, an alternative approach was developed (but not implemented) which integrates smoothly with the data-directed control flow of Hearsay-II.

This alternative approach is based on introducing the concept of a *goal node* into the blackboard, with types of attributes distinct from those of an hypothesis, and a means of relating goals at different levels. The action of creating a goal at a particular level is a monitorable event that triggers a KS that can do processing at that level. By making a goal node distinct from an hypothesis, a policy KS can generate goals without interfering with KSs that operate at that information level but that cannot respond to the goal. If the triggered KS cannot directly satisfy the goal, it can generate a subgoal, linked to the original goal, to generate data at another level which could be used by the KS to satisfy the original goal. In this way, a policy KS can interact with KSs in an anonymous and independent way. For example, if there is no KS to react to the goal, processing can still continue. In the same manner, if there is more than one KS that can respond to the goal (i.e., competing KSs), the scheduler can resolve this conflict without the need for any action by the KS that generated the goal. A goal node can also be used as a convenient place for a generator type of KS action to leave internal state information about how much and what type of further processing it can do in this area.

The other major reason for violating the independence criterion was based on an efficiency consideration. As previously mentioned, it is comparatively expensive to create an hypothesis on the blackboard. The cost of hypothesis creation is especially critical with a KS that can potentially generate a large number of hypotheses. For example, the syntax prediction KS (EXTEND, in C2) can create, based on a prediction from a single phrase hypothesis, several hundred word hypotheses. Each of these must then be processed by the word verifier KS (VERIFY) and verified or rejected. Before these hypotheses are verified they share almost identical structures. All but twenty, perhaps, will be rejected by VERIFY. To avoid the expense of expanding these as distinct blackboard word hypotheses, special data structures have been constructed to store the predicted words compactly; these data structures are then attached as an attribute of the phrase hypothesis. This example further illustrates the weakness in the current Hearsay-II implementation of efficiently representing and processing groups of hypotheses.

Uniformity of Control

Another issue associated with the data-directed control structure is the ease with which

different global control strategies can be explored. The uniform interface conventions used for specifying and activating KSs and preconditions, together with treating policy (strategy) KSs in the same way as other KSs makes the total system easy to modify and understand.

As part of the uniform convention for specifying each KS, non-procedural declarations are required which tell the system the type of pattern that triggers the KS and the type of action that can result from the activation of the KS. By separating the activation of a KS from its scheduling, it has been easy to introduce new global strategies by applying a new priority evaluation function to the information supplied by each KS. In addition, by allowing a policy KS to be able to trigger upon certain conditions that occur in the scheduling "data base" (such as the absence of any invoked KSs, or the lack of any invoked KSs above a certain priority level), it is possible to add different types of policy KSs into the system in a modular manner (e.g., WOSCTL in configuration C2).

In the initial specification of the Hearsay-II architecture, the approach required for focus of control was not well developed and represented one of the major conceptual problems which would determine the success of the design. As a result of work on this problem over the last three years, it is felt that the problem, though not completely solved, is now understood well enough so that it no longer represents a major obstacle to the effective use of the architecture. It is interesting to note that much of the discussion in preceding sections is based on a better understanding of what features need to be present in the architecture in order to efficiently support complex focus of control strategies.

Parallel Processing

One of the initial design goals of the Hearsay-II architecture was that it should be efficiently (and correctly) executable on a multiprocessor [Les75Pa and Fen75Mu]. In order to test the parallel processing capabilities of this architecture on an actual KS configuration, a multiprocessor simulation system was embedded in the multiprocess implementation of Hearsay-II. Each KS in this configuration was modified with the appropriate synchronization primitives.

The result of this simulation, which used an early version of the C1 configuration that was strictly bottom-up in its processing (because it did not include the SASS module), showed that effective parallelism factors of four to six could be achieved [Fen77Pa]. Unfortunately, there does not exist similar simulation data for a fully configured C1 or C2 configuration, both of which include top-down processing. However, it is expected that the C2 configuration would exhibit a much higher degree of parallelism, because KS interaction is more loosely-coupled and the system does a large amount of breadth-first type of search.

The parallelism factors of four to six that were achieved were less than expected. Further experiments were performed to determine the reason for these low factors. One of these experiments was to run the system with all uses of the synchronization primitives turned off. In this mode, the parallelism factors increased to fourteen. This dramatic increase is due to the fact that much superfluous synchronization was performed in each KS to maintain data consistency because no assumptions were made about how the blackboard was modified by other KSs. This superfluous synchronization, combined with synchronization primitives whose granularity of locking was too coarse, led to unnecessarily large areas of the blackboard being locked in order to maintain data consistency; this resulted both in significant interference among concurrently executing KS processes and a high system overhead

(between 50 and 100 percent) in order to support parallel processing. As with context validation (discussed above), this was a price paid for complete independence among KSs.

A surprising result was that system performance, in terms of accuracy, was as good with the synchronization disabled as its performance with the full synchronization. The explanation for this phenomenon is that the asynchronous, data-directed control of Hearsay-II is robust in the face of certain types of synchronization errors. For example, consider the normal activity sequence of a KS which involves first examining the blackboard, and then, based on the values read, modifying the blackboard. Suppose that between the time when the KS read the value of an attribute on the blackboard and when it modified the blackboard, the value of the attribute was changed; therefore, the modification was inconsistent with the current state of the blackboard data. However, because of the data-directed nature of KS activation, the changing of the attribute will probably trigger the same KS to be reinvoked to recalculate its original modification. Thus, the need is obviated for a KS, while executing, to lockout the areas of the blackboard it has read, in order to maintain the consistency of its modifications. In addition, other types of inconsistency can often be resolved because another KS with a different view of the problem will correct an incorrect hypothesis whether it resulted from a synchronization error, a mistake in the theory used by the KS, or from errorful data. Thus, this self-correcting nature of information flow among KSs, created through the use of a data-directed form of the hypothesize-and-test paradigm, in many cases obviates the need for explicit use of synchronization.[13]

CONCLUSIONS

The major conclusions on the use of the multi-level blackboard structure are the following:

1. The paradigm of viewing problem solving in terms of hypothesize-and-test actions distributed among distinct representations of the problem (where these representations form a hierarchy of abstractions) has been shown to be a computationally feasible approach to solving knowledge-intensive tasks. This paradigm also provides a convenient framework for structuring and applying knowledge. This has been demonstrated both by the successful application of the Hearsay-II architecture to the speech understanding task and also its adoption as an approach to problem-solving in a diverse set of other domains such as image understanding [Pra77Se], reading comprehension [Rum76To], protein-crystallographic analysis [Eng77Kn], signal understanding [Nii77Ru], and complex learning [Sol77Kn].
2. The representation of alternative hypotheses in an integrated manner on the blackboard has been shown to have positive aspects. In particular, the integrated representation avoids unnecessary recalculation and makes it easy to compute a global view of the current state of the problem solution, for the purpose of focussing. The problems still to be resolved arise because the integrated representation permits hypotheses to be used simultaneously in (shared by) multiple contexts (hypothesis

13 Another example of this self-correcting type of computational structure is a class of iterative refinement methods used to solve partial differential equations. This type of computational structure can be decomposed for multiprocessor implementation so as to avoid most explicit synchronization at the expense of more cycles to reach convergence [Bau76As]. This decomposition is accomplished by not requiring each point in the differential grid to be calculated based on the most up-to-date value of its neighboring points.

networks). Existing primitives for grouping alternative hypotheses are inefficient in space, and, more importantly, make it difficult to determine easily the different contexts that use a hypothesis; these primitives also do not provide a convenient framework for representing and determining the fact that two contexts have very similar hypothesis structures.

3. There are problems with the current formulation of a partial solution as a distributed network of hypotheses at different information levels. There is a basic confusion in the Hearsay-II implementation between the hypothesis in the network which is at the highest level of abstraction (interpretation) and the entire network. This confusion, combined with the problem of handling of multiple uses of a hypothesis, makes it difficult to perform some of the complex focus-of-attention strategies possible in the architecture.
4. The uniform structure of the blackboard at all information levels has turned out to be a very positive feature of the architecture. It has made it possible to integrate new KSs into the system easily and to develop a large set of utilities applicable to all KSs. It has also permitted numerous reimplementations of the internal structure of the blackboard without requiring KS modification.

The major conclusions on the uniform, asynchronous, data-directed control structure are the following:

5. The use of an implicit and uniform control structure for KS cooperation makes the system easy to modify and understand. The separation permitted between the invocation of a KS and its scheduling makes it convenient to implement a variety of scheduling policies without KS modification.
6. The overhead costs involved in implementing this type of control structure are acceptable for KSs which do moderate amounts of internal computation at each invocation (e.g., more than 1/10 second in the current implementation).
7. This control structure is not appropriate for domains in which the hypothesis credibility ratings are not selective enough to suggest strongly good paths to search.
8. The problem of focus of attention in this type of control environment, though not completely solved, is now understood well enough so that it no longers represents a major obstacle to the effective use of the architecture. The integrated representation of alternatives on the blackboard, which permits a global view of the current state of problem solution, and the data-directed control structure make it possible to quickly refocus attention to the appropriate places in the blackboard.
9. The initial attempt to have complete KS independence (in both a sequential and parallel processing environment) resulted in a significant amount of overhead, and thus seems not to be worth the cost. A more balanced approach, based on some knowledge about the type of processing done by other KSs in the configuration, has been more effective. This knowledge does not violate anonymity of KSs because it is based on a functional characterization of their activity and not on their "names". Using this approach, KS configurations are still highly modular (i.e., there has been no serious subroutine interaction problem) without paying the severe costs (in complexity of KS programming and execution time) of complete independence.
10. Parallel processing can be exploited effectively in this architecture. The techniques which are needed because of the errorful nature of the processing in this problem domain provide a form of processing which is also robust in the face of data inconsistency caused by not imposing complete synchronization among parallel processes. Thus, the overhead costs of the synchronization are reduced substantially, allowing effective use of parallelism.

ACKNOWLEDGMENTS

Raj Reddy has provided much of the vision and energy for this work, most of the central ideas in the Hearsay model, and much technical expertise in many of the knowledge sources in the Hearsay-II system. Richard Fennell and Rick Hayes-Roth have been particularly instrumental in formulating and testing the Hearsay-II architecture. All members of the CMU "speech group" have contributed to this work; their substantial efforts are gratefully acknowledged. Dan Corkhill, Mark Fox, Doug Lenat, Jack Mostow, Don McCracken, John McDermott, Allen Newell, Ed Riseman, and Elliot Soloway have made helpful suggestions for this paper.

APPENDIX -- CONFIGURATIONS OF KNOWLEDGE SOURCES

Configuration C1

The KSs of C1 (see Figure 1) are functionally described here briefly. The name given in parentheses following the name of the KS is the module in which it was embedded.

SEG (SEG) -- The SEG KS [Gol76Se] generated, from the digitized acoustic signal, a sequence of contiguous, variable-length segment hypotheses.

CSEG (PSYN) -- This KS [Sho76Ph] combined segment hypotheses into larger segment hypotheses. The stimulus frame was a sequence of three contiguous segment hypotheses; the action was to generate one or more new segment hypotheses, each of whose times lay within the time span of the three hypotheses in the stimulus frame. The precondition for this KS was triggered highly asynchronously -- whenever a new segment hypothesis was created. The KS was then invoked once for every pair of segment hypotheses immediately preceding and following the new one.

PSYN (PSYN) -- This KS [Sho76Ph] created phone hypotheses, based on segment hypotheses. The stimulus frame was also a sequence of three contiguous segment hypotheses; the action was to generate one or more phonetic hypotheses, again with times within the boundaries of the stimulus hypotheses. The comment above about asynchrony of execution of CSEG also holds for PSYN.

POM (POMOW) -- The POM KS [Smi76Wo] generated syllable hypotheses from phone hypotheses. The stimulus frame contained phone hypotheses that were classified as syllable nuclei; the action of the KS was to create syllable hypotheses based on the stimulus frame and adjacent segment hypotheses. The precondition for this KS was very complex because it made no assumptions about the order in which phone hypotheses would be created. Thus, the creation of a new phone hypothesis of any kind (syllable nucleus or other) triggered the precondition and caused an invocation of the KS for each nucleus hypothesis with which the new phone hypothesis might possibly interact.

MOW (POMOW) -- The MOW KS [Smi76Wo] generated word hypotheses from contiguous syllable hypotheses. The stimulus frame consisted of a newly-created syllable hypothesis; the output word hypotheses covered the same time as the stimulus hypothesis, but could also encompass syllable hypotheses on either side of the stimulus hypothesis (i.e., for multi-syllabic words.) If the stimulus hypothesis suggested a multi-syllabic word but the hypothesis for the other syllables did not exist, the word would not be hypothesized; however, if at some later time the required syllable hypothesis did appear, the KS would be triggered (by the new syllable) and the word hypothesized.

RECOG (SASS) -- This RECOGnition KS [Hay76Sy] used syntactic knowledge to generate

phrase hypotheses from contiguous word or phrase hypotheses. The precondition triggered on a new phrase or word hypothesis (or one with a changed rating). If the triggering hypothesis completed, with existing hypotheses, a phrase and the constituents were rated sufficiently high, the KS was invoked. This was a bottom-up parsing action.

PREDICT (SASS) -- The PREDICTion KS [Hay76Sy] used syntactic knowledge to generate a new phrase hypothesis, given another phrase hypothesis that was highly rated. This was essentially a "sidewise" or "outward" action.

RESPELL (SASS) -- This KS [Hay76Sy], given a predicted phrase hypothesis (i.e., one with no links to lower level hypotheses, either phrase or word) with a sufficiently high prediction rating, generated hypotheses of the constituents (words and/or phrases) of the predicted hypothesis. Thus, respelling drove processing downward, from predicted hypotheses towards the word level, so that predictions could ultimately be matched to acoustic data and verified or rejected.

POSTDICT (SASS) -- Given a weakly recognized or predicted phrase or word hypothesis, this KS [Hay76Sy] looked for other hypotheses that tended to confirm it. Such hypotheses were linked to the "postdicted" hypothesis, increasing its rating.

WOM (WOMOS) -- This KS [Cro76Wo] was triggered on new word hypotheses that were not linked to syllable hypotheses (i.e., ones that were generated "from above", by RESPELL or PREDICT). For each such hypothesis, it generated (via a dictionary lookup) expected syllable hypotheses which were likely to describe it.

MOS (WOMOS) -- The MOS KS [Cro76Wo], given a new syllable hypothesis, generated (via a dictionary lookup) a set of surface-phonemic hypotheses which described the syllable.

TIME (POSSE) -- This KS [Cro76Wo] responded to the creation of a new phone or surface-phonemic hypothesis and attempted to create a link between the new hypothesis and an existing hypothesis at the other level.

SEARCH (POSSE) -- This KS [Cro76Wo] responded to the creation of a new link between a phone hypothesis and a surface-phoneme hypothesis and attempted to create new links adjacent to the triggering one. Thus, TIME and SEARCH together incrementally built, through structural connections on the blackboard, a synchronization of a sequence of surface-phonemes representing a syllable with a sequence of lower-level, acoustically-based phones. The SEARCH KS was very complex in that it built up competing synchronizations (multiple interpretations); this was done with localized, incremental actions and while attempting to have the competing interpretations share maximal consistent sub-structures.

RPOL (RPOL) -- This policy KS [Hay76Hy] was responsible for propagating validity ratings. It triggered on the creation of an hypothesis, the establishment of a structural connection between two hypotheses, or the change of rating of an hypothesis. It calculated ratings for an hypothesis based on the values of KS-assigned attributes and the ratings of its structurally connected neighboring hypotheses.

FOCUS (FOCUS) --This policy KS imposed a global control strategy on the function of all other KSs in the system. It imposed this control through the setting of goal hypotheses which indicated to a KS both that it should attempt to generate particular types of hypotheses and also what internal criterion (thresholds) it should apply in order to generate such hypotheses.

The strategy implemented by this KS was based on a progressive enlarging of the search space of hypotheses as existing hypotheses prove fruitless; the idea behind this strategy is that one should open up the combinatorics in the search space only when absolutely necessary. The strategy was implemented by setting up initial goal hypotheses with very high criteria for hypothesis generation and then successively lowering these thresholds when the search stagnated.

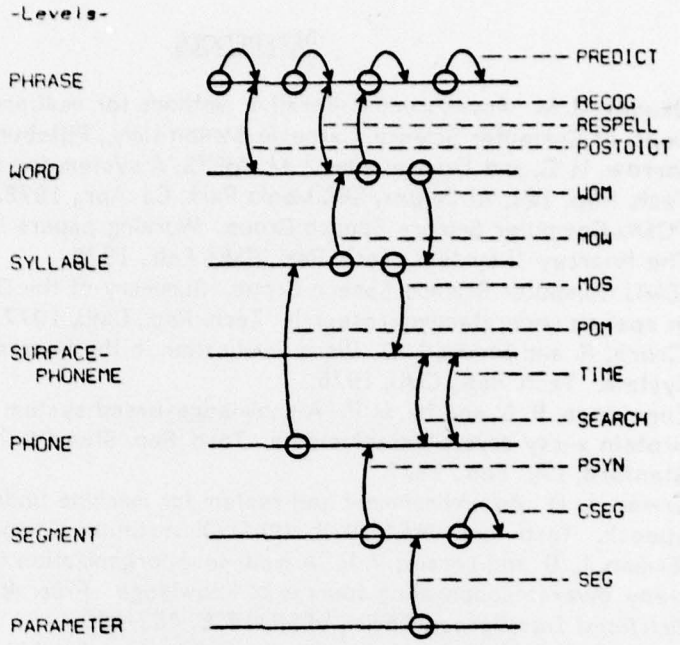


Figure 1. The levels and knowledge-sources of configuration C1. (As operational in January, 1976.)

Configuration C2

The KSs of C2 (see Figure 2) are described in the companion "Tutorial" paper.

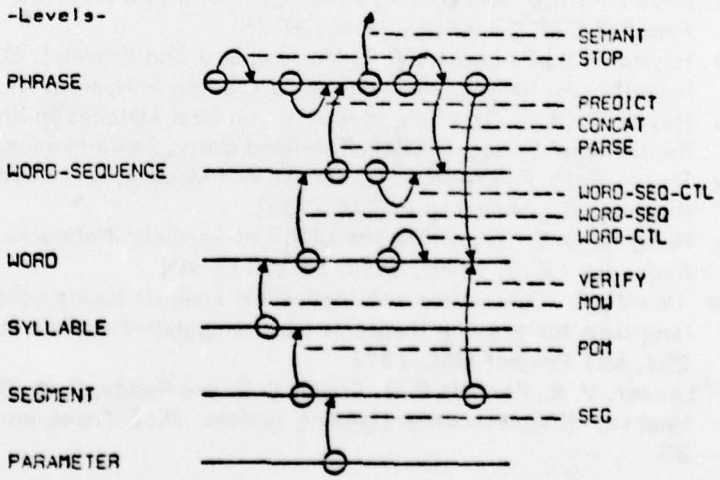


Figure 2. The levels and knowledge-sources of configuration C2. (As operational in September, 1976.)

REFERENCES

Bau76As Baudet, G. M. Asynchronous iterative methods for multiprocessors. Tech. Rep., Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, Pa, Nov., 1976.

Bar76Sc Barrow, H. G. and Tennenbaum, J. M. MSYS: A system for reasoning about scenes. Tech. Rep. 121, AI Center, SRI, Menlo Park, Ca, Apr., 1976.

CMU76W4 CMU Computer Science Speech Group. Working papers in speech recognition IV: The Hearsay-II system. Tech. Rep., CMU, Feb., 1976.

CMU77Su CMU Computer Science Speech Group. Summary of the CMU Five-year ARPA effort in speech understanding research. Tech. Rep., CMU, 1977.

Cro76Wo Cronk, R. and Erman, L. D. Word verification in the Hearsay-II speech understanding system. Tech. Rep., CMU, 1976.

Eng77Kn Engelmore, R. S. and Nii, H. P. A knowledge-based system for the interpretation of protein x-ray crystallographic data. Tech. Rep. Stan-CS-77-589, Stanford Univ., Stanford, CA. Feb., 1977.

Erm74En Erman, L. D. An environment and system for machine understanding of connected speech. Tech. Rep., CMU, 1974. (Ph.D. Dissertation, Comp. Sci. Dept., Stanford Univ.).

Erm75Mu Erman, L. D. and Lesser, V. R. A multi-level organization for problem solving using many diverse cooperating sources of knowledge. *Proc. 4th Inter. Joint Conf. on Artificial Intelligence*, Tbilisi, USSR, 1975, 483-490.

Fen75Mu Fennell, R. D. Multiprocess software architecture for AI problem solving. Tech. Rep., CMU, 1975. Ph.D. Dissertation.

Fen77Pa Fennell, R. D. and Lesser, V. R. Parallelism in AI problem solving: a case study of Hearsay-II. *IEEE Trans. on Computers*, C-26 (Feb., 1977), 98-111.

Fox77Ma Fox, M. S. and Mostow, D. J. Maximal Consistent Interpretations of Errorful Data in Hierarchically Modelled Domains. Tech. Rep., CMU, 1977.

Gol76Se Goldberg, H. G. Segmentation and labeling of connected speech. Appeared in [CM76W4].

Hay76Hy Hayes-Roth, F., Erman, L. D. and Lesser, V. R. Hypothesis validity ratings in the Hearsay-II speech understanding system. Appeared in [CM76W4].

Hay76Sy Hayes-Roth, F. and Mostow, D. J. Syntax and semantics in a distributed speech understanding system. *Proc. IEEE Inter. Conf. on Acoustics, Speech and Signal Processing*, Philadelphia, PA, 1976, 421-424. Also appeared in [CMU76W4].

Hay77Fo Hayes-Roth, F. and Lesser, V. R. Focus of attention in the Hearsay-II system. *Proc. 5IJCAI*, Cambridge, Mass., 27-35.

Hay77Po Hayes-Roth, F., Lesser, V. R., Mostow, D. J. and Erman, L. D. Policies for rating hypotheses, halting, and selecting a solution in Hearsay-II. Appeared in [CMU77Su].

Hay77Ro Hayes-Roth, F. The Role of Partial and Best Matches in Knowledge Systems. The Rand Paper Series, P-5802, The Rand Corp., Santa Monica, Ca., Jan., 1977.

Hay77Sy Hayes-Roth, F., Erman, L. D., Fox, M. and Mostow, D. J. Syntactic processing in Hearsay-II. Appeared in [CMU77Su].

Hen75Ex Hendrix, G. G. Expanding the Utility of Semantic Networks Through Partitioning. *Proc. 4th IJCAI*, Tbilisi, USSR, 1975, 115-121.

Hew72De Hewitt, C. Description and theoretical analysis (using schemata) of Planner: A language for proving theorems and manipulating models in a robot. AI Memo No. 251, MIT Project MAC, 1972.

Les75Or Lesser, V. R., Fennell, R. D., Erman, L. D. and Reddy, D. R. Organization of the Hearsay-II speech understanding system. *IEEE Trans. on ASSP 23* (Jan., 1975) 11-23.

Les75Pa Lesser, V. R. Parallel processing in speech understanding systems: a survey of design problems. In *Speech Recognition: Invited Papers of the IEEE Symp.*. (Reddy, D. R., Ed.) Academic Press, 1975, 481-499.

Les77Se Lesser, V. R., Hayes-Roth, F., Birnbaum, M. and Cronk, R. Selection of word islands in the Hearsay-II speech understanding system. *Proc. 1977 IEEE Inter. Conf. on ASSP*, Hartford, CT, 1977.

Low76Ha Lowerre, B. T. The Harpy speech recognition system. Tech. Rep., CMU, 1976. Ph.D. Dissertation.

McK77Wo McKeown, D. M. Word verification in the Hearsay-II speech understanding system. *Proc. 1977 IEEE Inter. Conf. on ASSP*. Hartford, Ct.

Min74Fr Minsky, M. A framework for representing knowledge. AI memo No. 306, MIT, June, 1974.

Mos77Ha Moslow, D. J. A halting condition and related pruning heuristic for combinatorial search. Tech. Rep., CMU, 1977.

New73Sp Newell, A., Barnett, J., Forgie, J., Green, C., Klatt, D., Licklider, J. C. R., Munson, J., Reddy, R. and Woods, W. *Speech Understanding Systems: Final Report of a Study Group*. North-Holland, 1973. (Originally appeared in 1971.)

Nii77Ru Nii, H. P. and Feigenbaum, E. A. Rule-based understanding of signals. Conf. on Pattern-Directed Inference Systems, Hawaii, 1977.

Pra77Se Prager, J., Nagin, P., Kohler, R., Hansen, A. and Riseman, E. Sementation processes in the VISIONS system. IJCAI-77, Cambridge, Mass., 1977.

Red73Mo Reddy, D. R., Erman, L. D. and Neely, R. B. A model and a system for machine recognition of speech. *IEEE Trans. on Audio and Electroacoustics, AU-21*, 1973, 229-238.

Red73Hx Reddy, D. R., Erman, L. D., Fennell, R. D. and Neely, R. B. The Hearsay speech understanding system: an example of the recognition process. *Proc. 3rd IJCAI*, Stanford, CA, 1973, 185-193.

Rei76SA Reiser, J. F. SAIL. Stanford Artificial Intelligence Laboratory, Memo AIM-289, 1976.

Ros76Sc Rosenfeld, A., Hummel, R. A. and Zucker, S. W. Scene labeling by relaxation operations. *IEEE Trans. Systems, Man and Cybernetics, SMC-6*, 420-433, 1976.

Rub77Lo Rubin, S. M. and Reddy, D. R. The LOCUS model of search and its use in image interpretation. *Proc. 5th IJCAI*, Cambridge, MA, 1977.

Rul73Qa Rulifson, J. F., et al. QA4: A procedural calculus for intuitive reasoning. Tech. Note 73, AI Center, SRI, Menlo Park, Ca, 1973.

Rum76To Rumelhart, D. E. Toward an interactive model of reading. Tech. Rep. 56, Center for Human Information Processing, Univ. of Cal. at San Diego, La Jolla, CA.

Sho76Ph Shockey, L. and Adam, C. The phonetic component of the Hearsay-II speech understanding system. In CM76W4.

Smi76Wo Smith, A. R. Word hypothesization in the Hearsay-II speech system. *Proc. IEEE Int. Conf. on ASSP*, Philadelphia, PA, 1976.

Sol77Kn Soloway, E. M. and Riseman, E. M. Knowledge-directed learning. Conf. on Pattern-Directed Inference Systems, Hawaii, 1977.

Wal77Ov Walker, D. et al. An overview of speech understanding research at SRI. *Proc. 5th IJCAI*, Cambridge, MA, 1977.

Woo76Fi Woods, W. A. et al. *Speech understanding systems, Final report*, Nov. 74 -- Oct. 76. BBN report 3438, Bolt Beranek and Newman, Inc., Cambridge, MA, 1976.